Lattice-based Cryptography*

Daniele Micciancio† Oded Regev‡

November 7, 2008

1 Introduction

In this chapter we describe some of the recent progress in *lattice-based cryptography*. Lattice-based cryptographic constructions hold a great promise for post-quantum cryptography, as they enjoy very strong security proofs based on worst-case hardness, relatively efficient implementations, as well as great simplicity. In addition, lattice-based cryptography is believed to be secure against quantum computers. Our focus here will be mainly on the practical aspects of lattice-based cryptography and less on the methods used to establish their security. For other surveys on the topic of lattice-based cryptography, see, e.g., [60, 36, 72, 51] and the lecture notes [50, 68]. The survey by Nguyen and Stern [60] also describes some applications of lattices in cryptanalysis, an important topic that we do not discuss here. Another useful resource is the book by Micciancio and Goldwasser [53], which also contains a wealth of information on the computational complexity aspects of lattice problems.

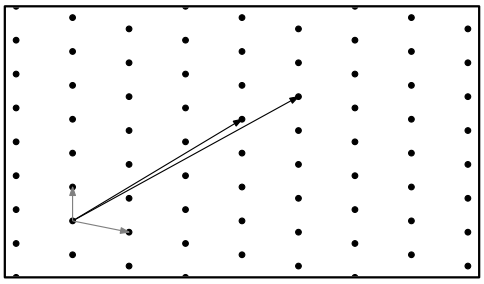

Figure 1: A two-dimensional lattice and two possible bases.

So what is a lattice? A lattice is a set of points in n-dimensional space with a periodic structure, such as the one illustrated in Figure 1. More formally, given n-linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$, the lattice generated by them is the set of vectors

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z} \right\}.$$

The vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are known as a *basis* of the lattice.

*An edited copy of this chapter appears in "Post Quantum Cryptography", D.J. Bernstein; J. Buchmann; E. Dahmen (eds.), pp. 147-191, Springer (February 2009). This is the authors' copy.

†CSE Department, University of California, San Diego, La Jolla, CA 92093, USA. Supported in part by NSF Grant CCF 0634909.

‡School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Supported by the Binational Science Foundation, by the Israel Science Foundation, and by the European Commission under the Integrated Project QAP funded by the IST directorate as Contract Number 015848.

The way lattices can be used in cryptography is by no means obvious, and was discovered in a breakthrough paper by Ajtai [4]. His result has by now developed into a whole area of research whose main focus is on expanding the scope of lattice-based cryptography and on creating more practical lattice-based cryptosystems. Before discussing this area of research in more detail, let us first describe the computational problems involving lattices, whose presumed hardness lies at the heart of lattice-based cryptography.

1.1 Lattice problems and algorithms

Lattice-based cryptographic constructions are based on the presumed hardness of lattice problems, the most basic of which is the *shortest vector problem* (SVP). Here, we are given as input a lattice represented by an arbitrary basis, and our goal is to output the shortest nonzero vector in it. In fact, one typically considers the approximation variant of SVP where the goal is to output a lattice vector whose length is at most some approximation factor $\gamma(n)$ times the length of the shortest nonzero vector, where n is the dimension of the lattice. A more precise definition of SVP and several other lattice problems appears in Section 2.

The most well known and widely studied algorithm for lattice problems is the *LLL algorithm*, developed in 1982 by Lenstra, Lenstra, and Lovász [39]. This is a polynomial time algorithm for SVP (and for most other basic lattice problems) that achieves an approximation factor of $2^{O(n)}$ where n is the dimension of the lattice. As bad as this might seem, the LLL algorithm is surprisingly useful, with applications ranging from factoring polynomials over the rational numbers [39], to integer programming [31], as well as many applications in cryptanalysis (such as attacks on knapsack-based cryptosystems and special cases of RSA).

In 1987, Schnorr presented an extension of the LLL algorithm leading to somewhat better approximation factors [74]. The main idea in Schnorr's algorithm is to replace the core of the LLL algorithm, which involves 2×2 blocks, with blocks of larger size. Increasing the block size improves the approximation factor (i.e., leads to shorter vectors) at the price of an increased running time. Schnorr's algorithm (e.g., as implemented in Shoup's NTL package [76]) is often used by experimenters. Several variants of Schnorr's algorithm exist, such as the recent one by Gama and Nguyen [15] which is quite natural and elegant. Unfortunately, all these variants achieve more or less the same exponential approximation guarantee.

If one insists on an *exact* solution to SVP, or even just an approximation to within $\mathrm{poly}(n)$ factors, the best known algorithm has a running time of $2^{O(n)}$ [7]. The space requirement of this algorithm is unfortunately also exponential which makes it essentially impractical (but see [61] for a recent implementation that can handle dimensions up to 50). Other algorithms require only polynomial space, but run in $2^{O(n \log n)}$ time (see [31] and the references in [61]).

The above discussion leads us to the following conjecture.

Conjecture 1.1 *There is no polynomial time algorithm that approximates lattice problems to within polynomial factors.*

Less formally, it is conjectured that approximating lattice problems to within polynomial factors is a hard problem (see also [73]). As we shall see later, the security of many lattice-based cryptographic constructions is based on this conjecture. As a further evidence for this conjecture, we note that progress in lattice algorithms has been stubbornly difficult, with no significant improvement in performance since the 1980s. This is in contrast to number theoretic problems such as factoring for which we have some remarkable subexponential time algorithms like the number field sieve [38]. We should note, though, that approximating lattice problems to within factors above $\sqrt{n/\log n}$ is *not* NP-hard unless the polynomial time hierarchy collapses [37, 19, 2]; NP-hardness results for lattice problems are known only for much smaller approximation factors such as $n^{O(1/\log\log n)}$ (see [78, 3, 12, 14, 48, 33, 25] and the survey [34]).

When applied to "real-life" lattices or lattices chosen randomly from some natural distribution, lattice reduction algorithms tend to perform somewhat better than their worst-case performance. This phenomenon is still not fully explained, but has been observed in many experiments. In one such recent investigation [16], Gama and Nguyen performed extensive experiments with several lattice reduction algorithms and several distributions on lattices. One of their conclusions is that known lattice reduction algorithms provide an approximation ratio of roughly δ^n where n is the dimension of the lattice and δ is a constant that depends

on the algorithm. The best δ achievable with algorithms running in reasonable time is very close to 1.012. Moreover, it seems that approximation ratios of $(1.01)^n$ are outside the reach of known lattice reduction algorithm. See Section 3 for a further discussion of the Gama-Nguyen experiments.

1.2 Lattice-based cryptography

As mentioned in the beginning of this chapter, lattice-based cryptographic constructions hold a great promise for post-quantum cryptography. Many of them are quite efficient, and some even compete with the best known alternatives; they are typically quite simple to implement; and of course, they are all believed to be secure against quantum computers (a topic which we will discuss in more detail in the next subsection).

In terms of security, lattice-based cryptographic constructions can be divided into two types. The first includes practical proposals, which are typically very efficient, but often lack a supporting proof of security. The second type admit strong provable security guarantees based on the worst-case hardness of lattice problems, but only a few of them are sufficiently efficient to be used in practice. We will consider both types in this chapter, with more emphasis on the latter type.

In the rest of this subsection, we elaborate on the strong security guarantees given by constructions of the latter type, namely that of worst-case hardness. What this means is that breaking the cryptographic construction (even with some small non-negligible probability) is provably at least as hard as solving several lattice problems (approximately, within polynomial factors) in the *worst case*. In other words, breaking the cryptographic construction implies an efficient algorithm for solving *any* instance of some underlying lattice problem. In most cases, the underlying problem is that of approximating lattice problems such as SVP to within polynomial factors, which as mentioned above, is conjectured to be a hard problem.

Such a strong security guarantee is one of the distinguishing features of lattice-based cryptography. Virtually all other cryptographic constructions are based on average-case hardness. For instance, breaking a cryptosystem based on factoring might imply the ability to factor *some* numbers chosen *according to a certain distribution*, but not the ability to factor *all* numbers

The importance of the worst-case security guarantee is twofold. First, it assures us that attacks on the cryptographic construction are likely to be effective only for small choices of parameters and not asymptotically. In other words, it assures us that there are no fundamental flaws in the design of our cryptographic construction. In fact, in some cases, the worst-case security guarantee can even guide us in making design decisions. Second, in principle the worst-case security guarantee can help us in choosing concrete parameters for the cryptosystem, although in practice this leads to what seems like overly conservative estimates, and as we shall see later, one often sets the parameters based on the best known attacks.

1.3 Quantum and lattices

As we have seen above, lattice problems are typically quite hard. The best known algorithms either run in exponential time, or provide quite bad approximation ratios. The field of lattice-based cryptography has been developed based on the assumption that lattice problems are hard. But is lattice-based cryptography suitable for a post-quantum world? Are lattice problems hard even for quantum computers?

The short answer to this is "probably yes": *There are currently no known quantum algorithms for solving lattice problems that perform significantly better than the best known classical (i.e., non-quantum) algorithms* (but see [41]). This is despite the fact that lattice problems seem like a natural candidate to attempt to solve using quantum algorithms: because they are believed not to be NP-hard for typical approximation factors, because of their periodic structure, and because the Fourier transform, which is used so successfully in quantum algorithms, is tightly related to the notion of lattice duality.

Attempts to solve lattice problems by quantum algorithms have been made since Shor's discovery of the quantum factoring algorithm in the mid-1990s, but have so far met with little success if any at all. The main difficulty is that the periodicity finding technique, which is used in Shor's factoring algorithm and related quantum algorithms, does not seem to be applicable to lattice problems. It is therefore natural to consider the following conjecture, which justifies the use of lattice-based cryptography for post-quantum cryptography:

Conjecture 1.2 *There is no polynomial time quantum algorithm that approximates lattice problems to within polynomial factors.*

The above discussion, however, should not be interpreted as saying that the advent quantum algorithms had no influence on our understanding of lattice problems. Although actual quantum algorithms for lattice problems are not known, there are a few very intriguing connections between quantum algorithms and lattice problems. The first such connection was demonstrated in [70] where it was shown that a certain extension of the period finding problem to non-Abelian groups can be used to give quantum algorithms for lattice problems. This approach, unfortunately, has so far not led to any interesting quantum algorithms for lattice problems.

A possibly more interesting connection is the use of a quantum hardness assumption in the lattice-based cryptosystem of [71]. A detailed discussion of this cryptosystem and its applications will appear in Subsection 5.4. For now, we briefly discuss the way quantum algorithms are used there. The main observation made there is that quantum algorithms *can* be useful in solving lattice problems, albeit somewhat unnatural ones. Consider the following scenario. We are given an oracle that is able to answer queries of the following type: on input a lattice $\mathcal{L}$ and a point $\mathbf{x}$ that is somewhat close to $\mathcal{L}$, it outputs the closest lattice point to $\mathbf{x}$. If $\mathbf{x}$ is not close enough to $\mathcal{L}$, the output of the oracle is undefined. In some sense, such an oracle seems quite powerful: the best known algorithms for performing such a task require exponential time. Nevertheless, there seems to be absolutely no way to do anything "useful" with this oracle classically! Indeed, it seems that the only way to generate inputs to the oracle is the following: somehow choose a lattice point $\mathbf{y} \in \mathcal{L}$ and let $\mathbf{x} = \mathbf{y} + \mathbf{z}$ for some small perturbation vector $\mathbf{z}$. We can now feed $\mathbf{x}$ to the oracle since it is close to the lattice. But the result we get, $\mathbf{y}$, is totally useless since we already know it!

It turns out that in the quantum setting, such an oracle is quite useful. Indeed, being able to compute $\mathbf{y}$ from $\mathbf{x}$ allows to *uncompute* $\mathbf{y}$. More precisely, it allows to transform the quantum state $|\mathbf{x}, \mathbf{y}\rangle$ to the state $|\mathbf{x}, 0\rangle$ in a reversible (i.e., unitary) way. This ability to erase the content of a memory cell in a reversible way seems useful only in the quantum setting. By using this together with the Fourier transform, it is shown in [71] how to use such an oracle in order to find short lattice vectors in the dual lattice.

1.4 Organization

The rest of this chapter is organized as follows. In Section 2 we provide some preliminaries on lattices. In Section 3 we consider a certain lattice problem that lies at the heart of many lattice-based cryptographic constructions, and discuss the best known algorithms for solving it. This will be used when we suggest concrete parameters for lattice-based constructions. The next three sections discuss three main cryptographic primitives: hash functions (Section 4), public key cryptosystems (Section 5), and digital signature schemes (Section 6). Some recent constructions of other cryptographic primitives are mentioned in Section 7. Finally, in Section 8 we list some of the main open questions in the area.

2 Preliminaries

All logarithms are base 2 unless otherwise indicated. We use column notation for vectors and use $(x_1, \ldots, x_n)$ to denote the column vector with entries $x_1, \ldots, x_n$. We use square brackets to enclose matrices and row vectors.

Lattices: A *lattice* is defined as the set of all integer combinations

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \text{ for } 1 \leq i \leq n \right\}$$

of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^n$ (see Figure 1). The set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a *basis* for the lattice. A basis can be represented by the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ having the basis

vectors as columns. Using matrix notation, the lattice generated by a matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ can be defined as $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \colon \mathbf{x} \in \mathbb{Z}^n\}$, where $\mathbf{Bx}$ is the usual matrix-vector multiplication.

It is not difficult to see that if $\mathbf{U}$ is a unimodular matrix (i.e., an integer square matrix with determinant ± 1), the bases $\mathbf{B}$ and $\mathbf{BU}$ generate the same lattice. (In fact, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if there exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{B}' = \mathbf{BU}$.) In particular, any lattice admits multiple bases, and this fact is at the core of many cryptographic applications.

The *determinant* of a lattice is the absolute value of the determinant of the basis matrix $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. The value of the determinant is independent of the choice of the basis, and geometrically corresponds to the inverse of the density of the lattice points in $\mathbb{R}^n$. The *dual* of a lattice $\mathcal{L}$ in $\mathbb{R}^n$, denoted $\mathcal{L}^*$, is the lattice given by the set of all vectors $\mathbf{y} \in \mathbb{R}^n$ satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$ for all vectors $\mathbf{x} \in \mathcal{L}$. It can be seen that for any $\mathbf{B} \in \mathbb{R}^{n \times n}$, $\mathcal{L}(\mathbf{B})^* = \mathcal{L}((\mathbf{B}^{-1})^T)$. From this it follows that $\det(\mathcal{L}^*) = 1/\det(\mathcal{L})$.

q-ary lattices: Of particular importance in lattice-based cryptography are *q-ary lattices*. These are lattices $\mathcal{L}$ satisfying $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ for some (possibly prime) integer q. In other words, the membership of a vector $\mathbf{x}$ in $\mathcal{L}$ is determined by $\mathbf{x} \bmod q$. Such lattices are in one-to-one correspondence with linear codes in $\mathbb{Z}_q^n$. Most lattice-based cryptographic constructions use q-ary lattices as their hard-on-average problem. We remark that any integer lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ is a q-ary lattice for some q, e.g., whenever q is an integer multiple of the determinant $\det(\mathcal{L})$. However, we will be mostly concerned with q-ary lattices with q much smaller than $\det(L)$.

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some integers q, m, n, we can define two m-dimensional q-ary lattices,

$$\Lambda_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \ : \ \mathbf{y} = \mathbf{A}^T\mathbf{s} \bmod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\}$$
$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \ : \ \mathbf{Ay} = \mathbf{0} \bmod q\}.$$

The first q-ary lattice is generated by the rows of $\mathbf{A}$; the second contains all vectors that are orthogonal modulo q to the rows of $\mathbf{A}$. In other words, the first q-ary lattice corresponds to the code generated by the rows of $\mathbf{A}$ whereas the second corresponds to the code whose parity check matrix is $\mathbf{A}$. It follows from the definition that these lattices are dual to each other, up to normalization; namely, $\Lambda_q^\perp(\mathbf{A}) = q \cdot \Lambda_q(\mathbf{A})^*$ and $\Lambda_q(\mathbf{A}) = q \cdot \Lambda_q^\perp(\mathbf{A})^*$.

Lattice problems: The most well known computational problems on lattices are the following.

- Shortest Vector Problem (SVP): Given a lattice basis $\mathbf{B}$, find the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$.
- Closest Vector Problem (CVP): Given a lattice basis $\mathbf{B}$ and a target vector $\mathbf{t}$ (not necessarily in the lattice), find the lattice point $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ closest to $\mathbf{t}$.
- Shortest Independent Vectors Problem (SIVP): Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$, find n linearly independent lattice vectors $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$ (where $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$ for all i) minimizing the quantity $\|\mathbf{S}\| = \max_i \|\mathbf{s}_i\|$.

In lattice-based cryptography, one typically considers the approximation variant of these problems, which are denoted by an additional subscript γ indicating the approximation factor. For instance, in SVP_γ the goal is to find a vector whose norm is at most γ times that of the shortest nonzero vector. Finally, let us mention that all problems can be defined with respect to any norm, but the Euclidean norm $\|\mathbf{x}\| = \sqrt{\sum_i x^2}$ is the most common (see [67]).

3 Finding Short Vectors in Random q-ary Lattices

Consider the following problem. We are given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some q, n and $m \geq n$ and we are asked to find a short vector in $\Lambda_q^\perp(\mathbf{A})$. What is the shortest vector that we can hope to find in a reasonable amount of time? Notice that this is equivalent to asking for a short solution to a set of n random

equations modulo q in m variables. There are two main methods to find such solutions, which we review in the next paragraphs. Before addressing the algorithmic question, let us try to estimate the length of the shortest nonzero vector. For this, assume q is prime. Then with high probability (assuming m is not too close to n), the rows of $\mathbf{A}$ are linearly independent over $\mathbb{Z}_q$. In such a case, the number of elements of $\mathbb{Z}_q^m$ that belong to $\Lambda_q^\perp(\mathbf{A})$ is exactly q^{m-n} from which it follows that $\det(\Lambda_q^\perp(\mathbf{A})) = q^n$. We can now heuristically estimate $\lambda_1(\Lambda_q^\perp(\mathbf{A}))$ as the smallest radius of a ball whose volume is q^n, i.e.,

$$\lambda_1(\Lambda_q^\perp(\mathbf{A})) \approx q^{n/m} \cdot ((m/2)!)^{1/m} / \sqrt{\pi} \approx q^{n/m} \cdot \sqrt{\frac{m}{2\pi e}}$$

where we used the formula for the volume of a ball in m dimensions. For reasonable values of m (that are not too close to n nor too large) this estimate seems to be very good, as indicated by some of our experiments in low dimensions. The above estimate applies if we are interested in vectors that have small Euclidean length. Similar arguments apply to other norms. For example, we can expect the lattice to contain nonzero vectors with coordinates all bounded in absolute value by

$$\lambda_1^\infty(\Lambda_q^\perp(\mathbf{A})) \approx \frac{q^{n/m} - 1}{2}.$$

Lattice reduction methods. We now get back to our original algorithmic question: what is the shortest vector that we can hope to find in a reasonable amount of time? In order to answer this question, we rely on the extensive experiments made by Gama and Nguyen in [16]. Although their experiments were performed on a different distribution on lattices, their results seem to apply very well also to the case of random q-ary lattices. Indeed, in all our experiments we observed the same behavior reported in their paper, with the exception that a "trivial" vector of length q can always be found; namely, the length of the vector obtained by running the best known algorithms on a random m-dimensional q-ary lattice $\Lambda_q^\perp(\mathbf{A})$ is close to

$$\min\{q, (\det(\Lambda_q^\perp(\mathbf{A})))^{1/m} \cdot \delta^m\} = \min\{q, q^{n/m}\delta^m\} \tag{1}$$

where the equality holds with high probability. The parameter δ depends on the algorithm used. Faster algorithms (which are unavoidable when the dimension is several hundreds) provide $\delta \approx 1.013$ whereas slower and more precise algorithms provide $\delta \approx 1.012$ or even $\delta \approx 1.011$. Lower values of δ seem to be impossible to obtain with our current understanding of lattice reduction. Gama and Nguyen in fact estimate that a factor of 1.005 is totally out of reach in dimension 500.

We now try to understand the effect that m has on the hardness of the question. A simple yet important observation to make is that the problem cannot become harder by increasing m. Indeed, we can always fix some of the variables (or coordinates) to 0 thereby effectively reducing to a problem with smaller m. In lattice terminology, this says that $\Lambda_q^\perp(\mathbf{A})$ contains as a "sublattice" $\Lambda_q^\perp(\mathbf{A}')$ where $\mathbf{A}'$ is obtained from $\mathbf{A}$ by removing some of its columns. (More precisely, since the two lattices are of different dimensions, we need to append zero coordinates to the latter in order for it to be a true sublattice of the former.)

In Figure 2 we plot $q^{n/m}\delta^m$ as a function of m. It is easy to see that the minimum of the function is $2^{2\sqrt{n \log q \log \delta}}$ and is obtained for $m = \sqrt{n \log q / \log \delta}$. This means that when applying lattice reduction algorithms to $\Lambda_q^\perp(\mathbf{A})$, the shortest vectors are produced when $m = \sqrt{n \log q / \log \delta}$. For smaller m, the lattice is too sparse and does not contain short enough vectors. For larger m, the high dimension prevents lattice reduction algorithms from finding short vectors. In such a case, one is better off removing some of the columns of $\mathbf{A}$ in order to arrive at a lower dimensional problem. We note that this phenomenon has showed up clearly in our experiments.

To summarize, based on the experiments made by Gama and Nguyen, we can conclude that the shortest vector one can find in $\Lambda_q^\perp(\mathbf{A})$ for a random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ using state of the art lattice reduction algorithms is of length at least

$$\min\{q, 2^{2\sqrt{n \log q \log \delta}}\}, \tag{2}$$

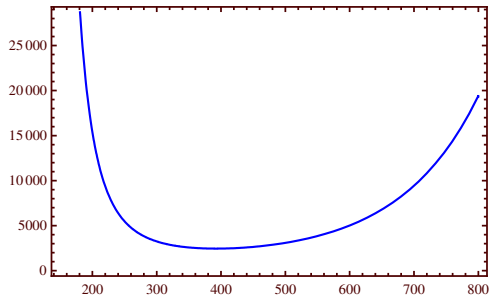

Figure 2: Estimated length of vector found with $\delta = 1.01$, $q = 4416857$, and $n = 100$ as a function of m.

where δ is not less than 1.01. Notice that the above expression is independent of m. This indicates that the difficulty of the problem depends mainly on n and q and not so much on m. Interestingly, the parameter m plays a minor role also in Ajtai's worst-case connection, giving further evidence that n and q alone determine the difficulty of the problem.

Combinatorial methods. It is interesting to consider also combinatorial methods to find short vectors in a q-ary lattice, as for certain choices of parameters these methods perform better than lattice reduction. The best combinatorial methods to find short vectors in q-ary lattices are variants of the algorithms presented [9, 79], e.g., as described in [45] in the context of attacking lattice-based hash functions.

The method works as follows. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, say we want to find a lattice point in $\Lambda_q^{\perp}(\mathbf{A})$ with coordinates all bounded in absolute value by b. We proceed as follows:

- Divide the columns of $\mathbf{A}$ into 2^k groups (for some k to be determined), each containing $m/2^k$ columns.
- For each group, build a list containing all linear combinations of the columns with coefficients in $\{-b, \ldots, b\}$.
- At this point we have 2^k lists, each containing $L = (2b+1)^{m/2^k}$ vectors in $\mathbb{Z}_q^n$. Combine the lists in pairs. When two lists are combined, take all the sums $\mathbf{x} + \mathbf{y}$ where $\mathbf{x}$ is an element of the first list, $\mathbf{y}$ is an element of the second list, and their sum $\mathbf{x} + \mathbf{y}$ is zero in the first $\log_q L$ coordinates. Since these coordinates can take $q^{\log_q L} = L$ values, we can expect the list resulting from the combination process to have size approximately equal to $L \cdot L/L = L$.
- At this point we have 2^{k-1} lists of size L containing vectors that are zero in their first $\log_q L$ coordinates. Keep combining the lists in pairs, until after k iterations we are left with only one list of size L containing vectors that are 0 in their first $k \cdot \log_q L$ coordinates. The parameter k is chosen in such a way that $n \approx (k+1) \log_q L$, or equivalently,

$$\frac{2^k}{k+1} \approx \frac{m \log(2b+1)}{n \log q}. \tag{3}$$

  For such a value of k, the vectors in the last list are zero in all but their last $n - k \log_q L \approx \log_q L$ coordinates. So, we can expect the list to contain the all zero vector.

The all zero vector found in the last list is given by a combination of the columns of $\mathbf{A}$ with coefficients bounded by b, so we have found the desired short lattice vector. Differently from lattice reduction, we can always expect this attack to succeed when $\mathbf{A}$ is random. The question is: what is the cost of running the attack? It is easy to see that the cost of the attack is dominated by the size of the lists L, which equals $(2b+1)^{m/2^k}$, where k is the largest integer satisfying (3). In certain settings (e.g., the construction of lattice-based hash functions presented in Section 4) lattice-based attacks stop finding short enough vectors well before the combinatorial attack becomes infeasible. So, the combinatorial attack can be used to determine the value of the parameters necessary to achieve a certain level of security.

Another difference between the combinatorial attack and those based on lattice reduction is that the combinatorial attack does take advantage of the large value of m. Larger values of m allow to use larger values for k, yielding shorter lists and more efficient attacks.

4 Hash Functions

A collision resistant hash function is a function $h : D \to R$ mapping a domain D to a much smaller set R, $|R| \ll |D|$ such that it is computationally hard to find collisions, i.e., input pairs $x_1, x_2 \in D$ such that $x_1 \neq x_2$ and still $h(x_1) = h(x_2)$. Technically, hash functions are often defined as keyed function families, where a collection of functions $\{h_k \colon D \to R\}$ is specified, and the security property is that given a randomly chosen k, no attacker can efficiently find a collision in h_k, even though such collisions certainly exist because D is larger than R. Collision resistant hash functions are very useful cryptographic primitives because they allow to compress a long message $x \in D$ to a short digest $h(x) \in R$, and still the digest is (computationally) bound to a unique x because of the collision resistance property.

For efficiency reasons, hash functions currently used in practice are based on ad-hoc design principles, similar to those used in the construction of block ciphers. Such functions, however, have been subject to attacks, raising interest in more theoretical constructions that can be proved secure based on some underlying mathematical problem. Collision resistant hash functions can be built starting from standard number theoretic problems (like the hardness of factoring integers, or the RSA problem), similar to those used in public key cryptography, but such constructions are unsatisfactory for two reasons: they are much slower than block ciphers, and they can be broken by quantum computers.

In this section we present various constructions of collision resistant hash functions based on lattices, starting from Ajtai's original work, and ending with SWIFFT, a highly efficient recent proposal based on a special class of lattices. These have several benefits over competing constructions: they admit supporting proofs of security (based on worst-case complexity assumptions), they appear to be resistant to quantum attacks, and the most efficient of them approaches efficiency levels comparable to those of traditional block cipher design. Finally, many techniques used in other lattice-based cryptographic constructions have been first developed in the context of collision resistant hashing. So, hash functions offer an excellent starting point to discuss the methods of lattice-based cryptography at large.

4.1 Ajtai's construction and further improvements

The first lattice-based cryptographic construction with worst-case security guarantees was presented in the seminal work of Ajtai [4]. Ajtai presented a family of one-way functions whose security is based on the worst-case hardness of n^c-approximate SVP for some constant $c > 0$. In other words, he showed that being able to invert a function chosen from this family with non-negligible probability implies the ability to solve *any* instance of n^c-approximate SVP.

Followup work concentrated on improving Ajtai's security proof. Goldreich et al. [20] showed that Ajtai's function is collision resistant, a stronger (and much more useful) security property than one-wayness. Most of the subsequent work focused on reducing the value of the constant c [11, 49, 54], thereby improving the security assumption. In the most recent work, the constant is essentially $c = 1$ [54]. We remark that all these constructions are based on the worst-case hardness of a problem not believed to be NP-hard (since $c \geq \frac{1}{2}$).

The main statement in all the above results is that for an appropriate choice of q, n, m, finding short vectors in $\Lambda_q^\perp(\mathbf{A})$ when $\mathbf{A}$ is chosen uniformly at random from $\mathbb{Z}_q^{n \times m}$ is as hard as solving certain lattice problems (such as approximate SIVP and approximate SVP) in the *worst case*. This holds even if the algorithm is successful in finding short vectors only with an inverse polynomially small probability (over the choice of matrix $\mathbf{A}$ and its internal randomness).

Once such a reduction is established, constructing a family of collision resistant hash functions is easy (see Algorithm 1). The hash function is parameterized by integers n, m, q, d. A possible choice is $d = 2$, $q = n^2$, and $m > n \log q / \log d$. The choice of n then determines the security of the hash function. The key to the hash

function is given by a matrix $\mathbf{A}$ chosen uniformly from $\mathbb{Z}_q^{n\times m}$. The hash function $f_{\mathbf{A}} : \{0,\dots,d-1\}^m \to \mathbb{Z}_q^n$ is given by $f_{\mathbf{A}}(\mathbf{y}) = \mathbf{A}\mathbf{y} \bmod q$. In terms of bits, the function maps $m\log d$ bits into $n\log q$ bits, hence we should choose $m > n\log q/\log d$ in order to obtain a hash function that compresses the input, or more typically $m \approx 2n\log q/\log d$ to achieve compression by a factor 2.

Notice that a collision $f_{\mathbf{A}}(\mathbf{y}) = f_{\mathbf{A}}(\mathbf{y}')$ for some $\mathbf{y} \neq \mathbf{y}'$ immediately yields a short non-zero vector $\mathbf{y}-\mathbf{y}' \in \Lambda_q^\perp(\mathbf{A})$. Using a worst-case to average-case reduction as above, we obtain that finding collisions for function $f_{\mathbf{A}}$ (even with an inverse polynomially small probability), is as hard as solving approximate SIVP and approximate SVP in the worst case.

Algorithm 1 A hash function following Ajtai's construction.

- **Parameters:** Integers $n, m, q, d \geq 1$.
- **Key:** A matrix $\mathbf{A}$ chosen uniformly from $\mathbb{Z}_q^{n\times m}$.
- **Hash function:** $f_{\mathbf{A}} : \{0,\dots,d-1\}^m \to \mathbb{Z}_q^n$ given by $f_{\mathbf{A}}(\mathbf{y}) = \mathbf{A}\mathbf{y} \bmod q$.

It is worth noting that this hash function is extremely simple to implement as it involves nothing but addition and multiplication modulo q, and q is a $O(\log n)$ bit number which comfortably fits into a single memory word or processor register. So, all arithmetic can be performed very efficiently without the need of the arbitrary precision integers commonly used in number theoretic cryptographic functions. As we shall see later, this is typical of lattice-based cryptography. Further optimizations can be obtained by choosing q to be a power of 2, and $d = 2$ which allows to represent the input as a sequence of m bits as well as to avoid the need for multiplications. Nevertheless, these hash functions are not particularly efficient because the key size grows at least quadratically in n. Consider for example setting $d = 2$, $q = n^2$, and $m = 2n\log q = 4n\log n$. The corresponding function has a key containing $mn = 4n^2\log n$ elements of $\mathbb{Z}_q$, and its evaluation requires roughly as many arithmetic operations. Collisions are given by vectors in $\Lambda_q^\perp(\mathbf{A})$ with entries in $\{1, 0, -1\}$. The combinatorial method described in Section 3 with bound $b = 1$ and parameter $k = 4$, yields an attack with complexity $L = 3^{m/16} \approx 2^{m/10}$. So, in order to get 100 bits of security ($L \approx 2^{100}$), one needs to set $m = 4n\log n \approx 1000$, and $n \geq 46$. This yields a hash function with a key size of $mn\log q \approx 500{,}000$ bits, and computation time of the order of $mn \approx 50{,}000$ arithmetic operations. Although still reasonable for a public key encryption function, this is considered unacceptable in practice for simpler cryptographic primitives like symmetric block ciphers or collision resistant hash functions.

4.2 Efficient hash functions based on cyclic and ideal lattices

The efficiency of lattice-based cryptographic functions can be substantially improved replacing general matrices by matrices with special structure. For example, in Algorithm 1, the random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ can be replaced by a block-matrix

$$\mathbf{A} = [\mathbf{A}^{(1)} \mid \dots \mid \mathbf{A}^{(m/n)}] \tag{4}$$

where each block $\mathbf{A}^{(i)} \in \mathbb{Z}_q^{n\times n}$ is a circulant matrix

$$\mathbf{A}^{(i)} = \begin{bmatrix} a_1^{(i)} & a_n^{(i)} & \cdots & a_3^{(i)} & a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \cdots & a_4^{(i)} & a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1}^{(i)} & a_{n-2}^{(i)} & \cdots & a_1^{(i)} & a_n^{(i)} \\ a_n^{(i)} & a_{n-1}^{(i)} & \cdots & a_2^{(i)} & a_1^{(i)} \end{bmatrix},$$

i.e., a matrix whose columns are all cyclic rotations of the first column $\mathbf{a}^{(i)} = (a_1^{(i)}, \ldots, a_n^{(i)})$. Using matrix notation, $\mathbf{A}^{(i)} = [\mathbf{a}^{(i)}, \mathbf{T}\mathbf{a}^{(i)}, \ldots, \mathbf{T}^{n-1}\mathbf{a}^{(i)}]$ where

$$\mathbf{T} = \left[\begin{array}{ccc|c} & \mathbf{0}^T & & 1 \\ \hline \ddots & & & \\ & \mathbf{I} & & \mathbf{0} \\ & & \ddots & \end{array}\right], \tag{5}$$

is the permutation matrix that rotates the coordinates of $\mathbf{a}^{(i)}$ cyclically. The circulant structure of the blocks has two immediate consequences:

- It reduces the key storage requirement from nm elements of $\mathbb{Z}_q$ to just m elements, because each block $\mathbf{A}^{(i)}$ is fully specified by its first column $\mathbf{a}^{(i)} = (a_1^{(i)}, \ldots, a_n^{(i)})$.
- It also reduces (at least asymptotically) the running time required to compute the matrix-vector product $\mathbf{A}\mathbf{y} \bmod q$, from $O(mn)$ arithmetic operations (over $\mathbb{Z}_q$), to just $\tilde{O}(m)$ operations, because multiplication by a circulant matrix can be implemented in $\tilde{O}(n)$ time using the Fast Fourier Transform.

Of course, imposing any structure on matrix $\mathbf{A}$, immediately invalidates the proofs of security [4, 11, 49, 54] showing that finding collisions on the average is at least as hard as approximating lattice problems in the worst case. A fundamental question that needs to be addressed whenever a theoretical construction is modified for the sake of efficiency, is if the modification introduces security weaknesses.

The use of circulant matrices in lattice-based cryptography can be traced back to the NTRU cryptosystem [29], which is described in Section 5. However, till recently no theoretical results were known supporting the use of structured matrices in lattice-based cryptography. Several years after Ajtai's worst-case connection for general lattices [4] and the proposal of the NTRU cryptosystem [29], Micciancio [52] discovered that the efficient one-way function obtained by imposing a circulant structure on the blocks of (4) can still be proved to be hard to invert on the average based on the worst-case hardness of approximating SVP, albeit only over a restricted class of lattices which are invariant under cyclic rotation of the coordinates. Interestingly, no better algorithms (than those for general lattices) are known to solve lattice problems for such cyclic lattices. So, it is reasonable to assume that solving lattice problems on these lattices is as hard as the general case.

Micciancio's adaptation [52] of Ajtai's worst-case connection to cyclic lattices is non-trivial. In particular, Micciancio could only prove that the resulting function is one-way (i.e., hard to invert), as opposed to collision resistant. In fact, collisions can be efficiently found: in [43, 62] it was observed that if each block $\mathbf{A}^{(i)}$ is multiplied by a constant vector $c_i \cdot \mathbf{1} = (c_i, \ldots, c_i)$, then the output of $f_{\mathbf{A}}$ is going to be a constant vector $c \cdot \mathbf{1}$ too. Since c can take only q different values, a collision can be found in time q (or even $O(\sqrt{q})$, probabilistically), which is typically polynomial in n. Similar methods were later used in [45] to find collisions in the compression function of LASH, a practical hash function proposal modeled after the NTRU cryptosystem. The existence of collisions for these functions demonstrates the importance of theoretical security proofs whenever a cryptographic construction is modified.

While one-way functions are not strong enough security primitives to be directly useful in applications, the results of [52] stimulated theoretical interest in the construction of efficient cryptographic functions based on structured lattices, leading to the use of cyclic (and other similarly structured) lattices in the design of many other more useful primitives [62, 43, 44, 42], as well as further investigation of lattices with algebraic structure [63]. In the rest of this section, we describe the collision resistant hash functions of [62, 43], and their most recent practical instantiation [45]. Other cryptographic primitives based on structured lattices are described in Sections 5, 6, and 7.

4.2.1 Collision resistance from ideal lattices

The problem of turning the efficient one-way function of [52] into a collision resistant function was independently solved by Peikert and Rosen [62], and Lyubashevsky and Micciancio [43] using different (but related)

methods. Here we follow the approach used in the latter work, which also generalizes the construction of [52, 62] based on circulant matrices, to a wider range of structured matrices, some of which admit very efficient implementations [45]. The general construction, shown in Algorithm 2, is parametrized by integers n, m, q, d and a vector $\mathbf{f} \in \mathbb{Z}^n$, and it can be regarded as a special case of Algorithm 1 with structured keys $\mathbf{A}$. In Algorithm 2, instead of choosing $\mathbf{A}$ at random from the set of *all* matrices, one sets $\mathbf{A}$ to a block-matrix as in Eq. (4) with structured blocks $\mathbf{A}^{(i)} = \mathbf{F}^*\mathbf{a}^{(i)}$ defined as

$$\mathbf{F}^*\mathbf{a}^{(i)} = [\mathbf{a}^{(i)}, \mathbf{F}\mathbf{a}^{(i)}, \ldots, \mathbf{F}^{n-1}\mathbf{a}^{(i)}] \qquad \text{where} \quad \mathbf{F} = \left[\begin{array}{c|c} \begin{array}{ccc} & \mathbf{0}^T & \\ \hline \ddots & & \\ & \mathbf{I} & \\ & & \ddots \end{array} & -\mathbf{f} \end{array}\right].$$

The circulant matrices discussed earlier are obtained as a special case by setting $\mathbf{f} = (-1, 0, \ldots, 0)$, for which $\mathbf{F} = \mathbf{T}$ is just a cyclic rotation of the coordinates. The complexity assumption underlying the function is that lattice problems are hard to approximate in the worst case over the class of lattices that are invariant under transformation $\mathbf{F}$ (over the integers). When $\mathbf{f} = (-1, 0, \ldots, 0)$, this is exactly the class of cyclic lattices, i.e., lattices that are invariant under cyclic rotation of the coordinates. For general $\mathbf{f}$, the corresponding lattices have been named *ideal* lattices in [43], because they can be equivalently characterized as *ideals* of the ring of modular polynomials $\mathbb{Z}[x]/\langle f(x)\rangle$ where $f(x) = x^n + f_n x^{n-1} + \cdots + f_1 \in \mathbb{Z}[x]$. As for the class of cyclic lattices, no algorithm is known that solves lattice problems on ideal lattices any better than on general lattices. So, it is reasonable to assume that solving lattice problems on ideal lattices is as hard as the general case.

Algorithm 2 Hash function based on ideal lattices.

- **Parameters:** Integers q, n, m, d with $n|m$, and vector $\mathbf{f} \in \mathbb{Z}^n$.
- **Key:** m/n vectors $\mathbf{a}_1, \ldots, \mathbf{a}_{m/n}$ chosen independently and uniformly at random in $\mathbb{Z}_q^n$.
- **Hash function:** $f_{\mathbf{A}} : \{0, \ldots, d-1\}^m \to \mathbb{Z}_q^n$ given by

$$f_{\mathbf{A}}(\mathbf{y}) = [\mathbf{F}^*\mathbf{a}_1 \mid \ldots \mid \mathbf{F}^*\mathbf{a}_{m/n}]\mathbf{y} \bmod q.$$

Even for arbitrary $\mathbf{f}$, the construction described in Algorithm 2 still enjoys the efficiency properties of the one-way function of [52]: keys are represented by just m elements of $\mathbb{Z}_q$, and the function can be evaluated with $\tilde{O}(m)$ arithmetic operations using the Fast Fourier Transform (over the complex numbers). As usual, collisions are short vectors in the lattice $\Lambda_q^\perp([\mathbf{F}^*\mathbf{a}_1 \mid \ldots \mid \mathbf{F}^*\mathbf{a}_{m/n}])$. But, are short vectors in these lattices hard to find? We have already seen that in general the answer to this question is no: when $\mathbf{f} = (-1, 0, \ldots, 0)$ short vectors (and collisions in the hash function) can be easily found in time $O(q)$. Interestingly, [43] proves that finding short vectors in $\Lambda_q^\perp([\mathbf{F}^*\mathbf{a}_1 \mid \ldots \mid \mathbf{F}^*\mathbf{a}_{m/n}])$ on the average (even with just inverse polynomial probability) is as hard as solving various lattice problems (such as approximate SVP and SIVP) in the worst case over ideal lattices, provided the vector $\mathbf{f}$ satisfies the following two properties:

- For any two unit vectors $\mathbf{u}, \mathbf{v}$, the vector $[\mathbf{F}^*\mathbf{u}]\mathbf{v}$ has small (say, polynomial in n, typically $O(\sqrt{n})$) norm.
- The polynomial $f(x) = x^n + f_n x^{n-1} + \cdots + f_1 \in \mathbb{Z}[x]$ is irreducible over the integers, i.e., it does not factor into the product of integer polynomials of smaller degree.

Notice that the first property is satisfied by the vector $\mathbf{f} = (-1, 0, \ldots, 0)$ corresponding to circulant matrices, because all the coordinates of $[\mathbf{F}^*\mathbf{u}]\mathbf{v}$ are bounded by 1, and hence $\|[\mathbf{F}^*\mathbf{u}]\mathbf{v}\| \leq \sqrt{n}$. However, the polynomial $x^n - 1$ corresponding to $\mathbf{f} = (-1, 0, \ldots, 0)$ is not irreducible because it factors into $(x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1)$, and this is why collisions can be efficiently found. So, $\mathbf{f} = (-1, 0, \ldots, 0)$ is not a good choice

to get collision resistant hash functions, but many other choices are possible. For example, some choices of $\mathbf{f}$ considered in [43] for which both properties are satisfied (and therefore, result in collision resistant hash functions with worst-case security guarantees) are

- $\mathbf{f} = (1, \ldots, 1) \in \mathbb{Z}^n$ where $n+1$ is prime, and
- $\mathbf{f} = (1, 0, \ldots, 0) \in \mathbb{Z}^n$ for n equal to a power of 2.

The latter choice turns out to be very convenient from an implementation point of view, as described in the next subsection. Notice how ideal lattices associated to vector $(1, 0, \ldots, 0)$ are very similar to cyclic lattices: the transformation $\mathbf{F}$ is just a cyclic rotation of the coordinates, with the sign of the coordinate wrapping around changed, and the blocks of $\mathbf{A}$ are just circulant matrices, but with the elements above the diagonal negated. This small change in the structure of matrix $\mathbf{A}$ has dramatic effects on the collision resistance properties of the resulting hash function: If the signs of the elements above the diagonals of the blocks is not changed, then collisions in the hash function can be easily found. Changing the sign results in hash functions for which finding collisions is provably as hard as the worst-case complexity of lattice approximation problems over ideal lattices.

4.2.2 The SWIFFT hash function

The hash function described in the previous section is quite efficient and can be computed asymptotically in $\tilde{O}(m)$ time using the Fast Fourier Transform over the complex numbers. However, in practice, this carries a substantial overhead. In this subsection we describe the SWIFFT family of hash functions proposed in [45]. This is essentially a highly optimized variant of the hash function described in the previous section, and is highly efficient *in practice*, mainly due to the use of the FFT in $\mathbb{Z}_q$.

We now proceed to describe the SWIFFT hash function. As already suggested earlier, the vector $\mathbf{f}$ is set to $(1, 0, \ldots, 0) \in \mathbb{Z}^n$ for n equal to a power of 2, so that the corresponding polynomial $x^n + 1$ is irreducible. The novelty in [45] is a clever choice of the modulus q and a pre/post-processing operation applied to the key and the output of the hash function. More specifically, let q be a prime number such that $2n$ divides $q-1$, and let $\mathbf{W} \in \mathbb{Z}_q^{n \times n}$ be an invertible matrix over $\mathbb{Z}_q$ to be chosen later. The SWIFFT hash function maps a key $\tilde{\mathbf{a}}^{(1)}, \ldots, \tilde{\mathbf{a}}^{(m/n)}$ consisting of m/n vectors chosen uniformly from $\mathbb{Z}_q^n$ and an input $\mathbf{y} \in \{0, \ldots, d-1\}^m$ to $\mathbf{W} \cdot f_{\mathbf{A}}(\mathbf{y}) \bmod q$ where $\mathbf{A} = [\mathbf{F}^* \mathbf{a}^{(1)}, \ldots, \mathbf{F}^* \mathbf{a}^{(m/n)}]$ is as before and $\mathbf{a}^{(i)} = \mathbf{W}^{-1} \tilde{\mathbf{a}}^{(i)} \bmod q$. As we shall see later, SWIFFT can be computed very efficiently (even though at this point its definition looks more complicated than that of $f_{\mathbf{A}}$).

Notice that multiplication by the invertible matrix $\mathbf{W}^{-1}$ maps a uniformly chosen $\tilde{\mathbf{a}} \in \mathbb{Z}_q^n$ to a uniformly chosen $\mathbf{a} \in \mathbb{Z}_q^n$. Moreover, $\mathbf{W} \cdot f_{\mathbf{A}}(\mathbf{y}) = \mathbf{W} \cdot f_{\mathbf{A}}(\mathbf{y}') \pmod q$ if and only if $f_{\mathbf{A}}(\mathbf{y}) = f_{\mathbf{A}}(\mathbf{y}') \pmod q$. Together, these two facts establish that finding collisions in SWIFFT is equivalent to finding collisions in the underlying ideal lattice function $f_{\mathbf{A}}$, and the claimed collision resistance property of SWIFFT is supported by the connection [43] to worst case lattice problems on ideal lattices.

Algorithm 3 The SWIFFT hash function.

- **Parameters:** Integers n, m, q, d such that n is a power of 2, q is prime, $2n|(q-1)$ and $n|m$.
- **Key:** m/n vectors $\tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_{m/n}$ chosen independently and uniformly at random in $\mathbb{Z}_q^n$.
- **Input:** m/n vectors $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m/n)} \in \{0, \ldots, d-1\}^n$.
- **Output:** the vector $\sum_{i=1}^{m/n} \tilde{\mathbf{a}}^{(i)} \odot (\mathbf{W}\mathbf{y}^{(i)}) \in \mathbb{Z}_q^n$, where $\odot$ is the component-wise vector product.

We now explain the efficient implementation of SWIFFT given in Algorithm 3. By our choice of q, the multiplicative group $\mathbb{Z}_q^*$ of the integers modulo q has an element ω of order $2n$. Let

$$\mathbf{W} = [\omega^{(2i-1)(j-1)}]_{i=1,j=1}^{n,n}$$

n	m	q	d	ω	key size (bits)	input size (bits)	output size (bits)
64	1024	257	2	42	8192	1024	513

Table 1: Concrete parameters for the SWIFFT hash function achieving 100 bits of security.

be the Vandermonde matrix of $\omega, \omega^3, \omega^5, \ldots, \omega^{2n-1}$. Since ω has order $2n$, the elements $\omega, \omega^3, \omega^5, \ldots, \omega^{2n-1}$ are distinct, and hence the matrix $\mathbf{W}$ is invertible over $\mathbb{Z}_q$ as required. Moreover, it is not difficult to see that for any vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$, the identity

$$\mathbf{W}([\mathbf{F}^*\mathbf{a}]\mathbf{b}) = (\mathbf{W}\mathbf{a}) \odot (\mathbf{W}\mathbf{b}) \bmod q$$

holds true, where $\odot$ is the component-wise vector product. This implies that Algorithm 3 correctly computes

$$\mathbf{W} \cdot f_{\mathbf{A}}(\mathbf{y}) = \sum_{i=1}^{m/n} \mathbf{W}[\mathbf{F}^*\mathbf{a}^{(i)}]\mathbf{y}^{(i)} = \sum_{i=1}^{m/n} \tilde{\mathbf{a}}^{(i)} \odot (\mathbf{W}\mathbf{y}^{(i)}).$$

The most expensive part of the algorithm is the computation of the matrix-vector products $\mathbf{W}\mathbf{y}^{(i)}$. These can be efficiently computed using the FFT over $\mathbb{Z}_q$ as follows. Remember that the FFT algorithm over a field $\mathbb{Z}_q$ with an nth root of unity ζ (where n is a power of 2) allows to evaluate any polynomial $p(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{Z}_q[x]$ at *all* nth roots of unity ζ^i (for $i = 0, \ldots, n-1$) with just $O(n \log n)$ arithmetic operations in $\mathbb{Z}_q$. Using matrix notation and $\zeta = \omega^2$, the FFT algorithm computes the product $\mathbf{V}\mathbf{c}$ where $\mathbf{V} = [\omega^{2(i-1)(j-1)}]_{i,j}$ is the Vandermonde matrix of the roots $\omega^0, \omega^2, \ldots, \omega^{2(n-1)}$, and $\mathbf{c} = (c_0, \ldots, c_{n-1})$. Going back to the SWIFFT algorithm, the matrix $\mathbf{W}$ can be factored as the product $\mathbf{W} = \mathbf{V}\mathbf{D}$ of $\mathbf{V}$ by the diagonal matrix $\mathbf{D}$ with entries $d_{j,j} = \omega^{j-1}$. So, the product $\mathbf{W}\mathbf{y}^{(i)} = \mathbf{V}\mathbf{D}\mathbf{y}^{(i)}$ can be efficiently evaluated by first computing $\mathbf{D}\mathbf{y}^{(i)}$ (i.e., multiplying the elements of $\mathbf{y}^{(i)}$ component-wise by the diagonal of $\mathbf{D}$), and then applying the FFT algorithm over $\mathbb{Z}_q$ to $\mathbf{D}\mathbf{y}^{(i)}$ to obtain $\mathbf{W}\mathbf{y}^{(i)}$.

Several other implementation-level optimizations are possible, including the use of look-up tables and SIMD (single instruction multiple data) operations in the FFT computation. An optimized implementation of SWIFFT for the choice of parameters given in Table 1 is given in [45], which achieves throughput comparable to the SHA-2 family of hash functions.

Choice of parameters and security. The authors of [45] propose the set of parameters shown in Table 1. It is easy to verify that $q = 257$ is a prime, $2n = 128$ divides $q-1 = 256$, $n = 64$ divides $m = 1024$, $\omega = 42$ has order $2n = 128$ in $\mathbb{Z}_q^n$, and the resulting hash function $f_{\mathbf{A}} \colon \{0,1\}^m \to \mathbb{Z}_q^n$ has compression ratio approximately equal to 2, mapping $m = 1024$ input bits to one of $q^n = (2^8+1)^{64} < 2^{513}$ possible outputs. An issue to be addressed is how to represent the vector in $\mathbb{Z}_q^n$ output by SWIFFT as a sequence of bits. The easiest solution is to represent each element of $\mathbb{Z}_q$ as a sequence of 9 bits, so that the resulting output has $9 \cdot 64 = 576$ bits. It is also easy to reduce the output size closer to 513 bits at very little cost. (See [45] for details.)

We now analyze the security of SWIFFT with respect to combinatorial and lattice-based attacks. The combinatorial method described in Section 3 with bound $b = 1$ and parameter $k = 4$ set to the largest integer satisfying (3), yields an attack with complexity $L = 3^{m/16} \geq 2^{100}$.

Let us check that lattice-based attacks are also not likely to be effective in finding collisions. Collisions in SWIFFT are vectors in the m-dimensional lattice $\Lambda_q^{\perp}([\mathbf{F}^*\mathbf{a}_1 \mid \ldots \mid \mathbf{F}^*\mathbf{a}_{m/n}])$ with coordinates in $\{1, 0, -1\}$. Such vectors have Euclidean length at most $\sqrt{m} = 32$. However, according to estimate (2) for $\delta = 1.01$, state of the art lattice reduction algorithms will not be able to find nontrivial lattice vectors of Euclidean length bounded by

$$2^{2\sqrt{n \log q \log \delta}} \approx 42.$$

So, lattice reduction algorithms are unlikely to find collisions. In order to find lattice vectors with Euclidean length bounded by 32, one would need lattice reduction algorithms achieving $\delta < 1.0085$, which seems out of reach with current techniques, and even such algorithms would find vectors with short Euclidean length, but coordinates not necessarily in $\{1, 0, -1\}$.

5 Public Key Encryption Schemes

Several methods have been proposed to build public key encryption schemes based on the hardness of lattice problems. Some are mostly of theoretical interest, as they are still too inefficient to be used in practice, but admit strong provable security guarantees similar to those discussed in Section 4 for hash functions: breaking the encryption scheme (on the average, when the key is chosen at random) can be shown to be at least as hard as solving several lattice problems (approximately, within polynomial factors) in the *worst case*. Other schemes are practical proposals, much more efficient than the theoretical constructions, but often lacking a supporting proof of security.

In this section we describe the main lattice-based public key encryption schemes that have been proposed so far. We start from the GGH cryptosystem, which is perhaps the most intuitive encryption scheme based on lattices. We remark that the GGH cryptosystem has been subject to cryptanalytic attacks [58] even for moderately large values of the security parameter, and should be considered insecure from a practical point of view. Still, many of the elements of GGH and its HNF variant [47], can be found in other lattice-based encryption schemes. So, due to its simplicity, the GGH/HNF cryptosystem still offers a good starting point for the discussion of lattice-based public key encryption. Next, we describe the NTRU cryptosystem, which is the most practical lattice-based encryption scheme known to date. Unfortunately, neither GGH nor NTRU is supported by a proof of security showing that breaking the cryptosystem is at least as hard as solving some underlying lattice problem; they are primarily practical proposals aimed at offering a concrete alternative to RSA or other number theoretic cryptosystems.

The rest of this section is dedicated to theoretical constructions of cryptosystems that can be proved to be as hard to break as solving certain lattice problems in the worst case. We briefly review the Ajtai-Dwork cryptosystem (which was the first of its kind admitting a proof of security based on worst-case hardness assumptions on lattice problems) and followup work, and then give a detailed account of a cryptosystem of Regev based on a certain learning problem (called "learning with errors", LWE) that can be related to worst-case lattice assumptions via a quantum reduction. This last cryptosystem is currently the most efficient construction admitting a known theoretical proof of security. While still not as efficient as NTRU, it is the first theoretical construction approaching performance levels that are reasonable enough to be used in practice. Moreover, due to its algebraic features, the LWE cryptosystem has been recently used as the starting point for the construction of various other cryptographic primitives, as discussed in Section 7.

We remark that all cryptosystems described in this section are aimed at achieving the basic security notion called *semantic security* or *indistinguishability under chosen plaintext attack* [23]. This is a strong security notion, but only against passive adversaries that can intercept and observe (but not alter) ciphertexts being transmitted. Informally, semantic security means that an adversary that observes the ciphertexts being sent, cannot extract any (even partial) information about the underlying plaintexts (not even determining whether two given ciphertexts encrypt the same message) under any message distribution. Encryption schemes with stronger security guarantees (against active adversaries) are discussed in Section 7.

5.1 The GGH/HNF public key cryptosystem

The GGH cryptosystem, proposed by Goldreich, Goldwasser, and Halevi in [22], is essentially a lattice analogue of the McEliece cryptosystem [46] proposed 20 years earlier based on the hardness of decoding linear codes over finite fields. The basic idea is very simple and appealing. At a high level, the GGH cryptosystem works as follows:

- The private key is a "good" lattice basis $\mathbf{B}$. Typically, a good basis is a basis consisting of short, almost orthogonal vectors. Algorithmically, good bases allow to efficiently solve certain instances of the closest vector problem in $\mathcal{L}(\mathbf{B})$, e.g., instances where the target is very close to the lattice.
- The public key $\mathbf{H}$ is a "bad" basis for the same lattice $\mathcal{L}(\mathbf{H}) = \mathcal{L}(\mathbf{B})$. In [47], Micciancio proposed to use, as the public basis, the Hermite Normal Form (HNF) of $\mathbf{B}$. This normal form gives a lower[1]

[1]The HNF can be equivalently defined using upper triangular matrices. The choice between the lower or upper triangular formulation is pretty much arbitrary.

triangular basis for $\mathcal{L}(\mathbf{B})$ which is essentially unique, and can be efficiently computed from *any* basis of $\mathcal{L}(\mathbf{B})$ using an integer variant of the Gaussian elimination algorithm.[2] Notice that any attack on the HNF public key can be easily adapted to work with any other basis $\mathbf{B}'$ of $\mathcal{L}(\mathbf{B})$ by first computing $\mathbf{H}$ from $\mathbf{B}'$. So, in a sense, $\mathbf{H}$ is the worst possible basis for $\mathcal{L}(\mathbf{B})$ (from a cryptanalyst's point of view), and makes a good choice as a public basis.

- The encryption process consists of adding a short noise vector $\mathbf{r}$ (somehow encoding the message to be encrypted) to a properly chosen lattice point $\mathbf{v}$. In [47] it is proposed to select the vector $\mathbf{v}$ such that all the coordinates of $(\mathbf{r}+\mathbf{v})$ are reduced modulo the corresponding element along the diagonal of the HNF public basis $\mathbf{H}$. The vector $(\mathbf{r}+\mathbf{v})$ resulting from such a process is denoted $\mathbf{r} \bmod \mathbf{H}$, and it provably makes cryptanalysis hardest because $\mathbf{r} \bmod \mathbf{H}$ can be efficiently computed from any vector of the form $(\mathbf{r}+\mathbf{v})$ with $\mathbf{v} \in \mathcal{L}(\mathbf{B})$. So, any attack on $\mathbf{r} \bmod \mathbf{H}$ can be easily adapted to work on any vector of the form $\mathbf{r}+\mathbf{v}$ by first computing $(\mathbf{r}+\mathbf{v}) \bmod \mathbf{H} = \mathbf{r} \bmod \mathbf{H}$. Notice that $\mathbf{r} \bmod \mathbf{H}$ can be computed directly from $\mathbf{r}$ and $\mathbf{H}$ (without explicitly computing $\mathbf{v}$) by iteratively subtracting multiples of the columns of $\mathbf{H}$ from $\mathbf{r}$. Column $\mathbf{h}_i$ is used to reduce the ith element of $\mathbf{r}$ modulo $h_{i,i}$.
- The decryption problem corresponds to finding the lattice point $\mathbf{v}$ closest to the target ciphertext $\mathbf{c} = (\mathbf{r} \bmod \mathbf{H}) = \mathbf{v}+\mathbf{r}$, and the associated error vector $\mathbf{r} = \mathbf{c}-\mathbf{v}$.

The correctness of the GGH/HNF cryptosystem rests on the fact that the error vector $\mathbf{r}$ is short enough so that the lattice point $\mathbf{v}$ can be recovered from the ciphertext $\mathbf{v}+\mathbf{r}$ using the private basis $\mathbf{B}$, e.g., by using Babai's rounding procedure [8], which gives

$$\mathbf{v} = \mathbf{B}\lfloor \mathbf{B}^{-1}(\mathbf{v}+\mathbf{r}) \rceil.$$

On the other hand, the security relies on the assumption that without knowledge of a special basis (that is, given only the worst possible basis $\mathbf{H}$), solving these instances of the closest vector problem in $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{H})$ is computationally hard. We note that the system described above is *not* semantically secure because the encryption process is deterministic (and thus one can easily distinguish between ciphertexts corresponding to two fixed messages). In practice, one can randomly pad the message in order to resolve this issue (as is often done with the RSA function), although this is not rigorously justified.

Clearly, both the correctness and security depend critically on the choice of the private basis $\mathbf{B}$ and error vector $\mathbf{r}$. Since GGH has been subject to practical attacks, we do not review the specifics of how $\mathbf{B}$ and $\mathbf{r}$ were selected in the GGH cryptosystem, and move on to the description of other cryptosystems.

We remark that no asymptotically good attack to GGH is known: known attacks break the cryptosystem in practice for moderately large values of the security parameter, and can be avoided by making the security parameter even bigger. This, however, makes the cryptosystem impractical. The source of impracticality is similar to that affecting Ajtai's hash function discussed in the previous section, and can be addressed by similar means: general lattice bases require $\Omega(n^2)$ storage, and consequently the encryption/decryption running times also grow quadratically in the security parameter. As we will see shortly, much more efficient cryptosystems can be obtained using lattices with special structure, which admit compact representation.

5.2 The NTRU public key cryptosystem

NTRU is a ring-based cryptosystem proposed by Hoffstein, Pipher and Silverman in [29], which can be equivalently described using lattices with special structure. Below we present NTRU as an instance of the general GGH/HNF framework [22, 47] described in the previous subsection. We remark that this is quite different from (but still equivalent to) the original description of NTRU, which, in fact, was proposed concurrently to, and independently from [22].

Using the notation from Section 4, we let $\mathbf{T}$ be the linear transformation in Eq. (5) that rotates the coordinates of the input vector cyclically, and define $\mathbf{T}^*\mathbf{v} = [\mathbf{v}, \mathbf{T}\mathbf{v}, \ldots, \mathbf{T}^{n-1}\mathbf{v}]$ to be the circulant matrix of vector $\mathbf{v} \in \mathbb{Z}^n$. The lattices used by NTRU, named convolutional modular lattices in [29], are lattices in even

[2] Some care is required to prevent the matrix entries from becoming too big during intermediate steps of the computation.

dimension $2n$ satisfying the following two properties. First, they are closed under the linear transformation that maps the vector $(\mathbf{x}, \mathbf{y})$ (where $\mathbf{x}$ and $\mathbf{y}$ are n-dimensional vectors) to $(\mathbf{Tx}, \mathbf{Ty})$, i.e., the vector obtained by rotating the coordinates of $\mathbf{x}$ and $\mathbf{y}$ cyclically in parallel. Second, they are q-ary lattices, in the sense that they always contain $q\mathbb{Z}^{2n}$ as a sublattice, and hence membership of $(\mathbf{x}, \mathbf{y})$ in the lattice only depends on $(\mathbf{x}, \mathbf{y})$ mod q. The system parameters are a prime dimension n, an integer modulus q, a small integer p, and an integer weight bound d_f. For concreteness, we follow the latest NTRU parameter set recommendations [28], and assume q is a power of 2 (e.g., $q = 2^8$) and $p = 3$. More general parameter choices are possible, some of which are mentioned in [28], and we refer the reader to that publication and the NTRU Cryptosystems web site for details. The NTRU cryptosystem (described by Algorithm 4) works as follows:

- **Private Key.** The private key in NTRU is a short vector $(\mathbf{f}, \mathbf{g}) \in \mathbb{Z}^{2n}$. The lattice associated to a private key $(\mathbf{f}, \mathbf{g})$ (and system parameter q) is $\Lambda_q((\mathbf{T}^*\mathbf{f}, \mathbf{T}^*\mathbf{g})^T)$, which can be easily seen to be the smallest convolutional modular lattice containing $(\mathbf{f}, \mathbf{g})$. The secret vectors $\mathbf{f}, \mathbf{g}$ are subject to the following technical restrictions:
  - the matrix $[\mathbf{T}^*\mathbf{f}]$ should be invertible modulo q,
  - $\mathbf{f} \in \mathbf{e}_1 + \{p, 0, -p\}^n$ and $\mathbf{g} \in \{p, 0, -p\}^n$ are randomly chosen polynomials such that $\mathbf{f} - \mathbf{e}_1$ and $\mathbf{g}$ have exactly $d_f + 1$ positive entries and d_f negative ones. (The remaining $N - 2d_f - 1$ entries will be zero.)

  The bounds on the number of nonzero entries in $\mathbf{f} - \mathbf{e}_1$ and $\mathbf{g}$ are mostly motivated by efficiency reasons. More important are the requirements on the invertibility of $[\mathbf{T}^*\mathbf{f}]$ modulo q, and the restriction of $\mathbf{f} - \mathbf{e}_1$ and $\mathbf{g}$ to the set $\{p, 0, -p\}^n$, which are used in the public key computation, encryption and decryption operations. Notice that under these restrictions $[\mathbf{T}^*\mathbf{f}] \equiv \mathbf{I} \pmod{p}$ and $[\mathbf{T}^*\mathbf{g}] \equiv \mathbf{O} \pmod{p}$ (where $\mathbf{O}$ denotes the all zero matrix).

- **Public Key.** Following the general GGH/HNF framework, the NTRU public key corresponds to the HNF basis of the convolutional modular lattice $\Lambda_q((\mathbf{T}^*\mathbf{f}, \mathbf{T}^*\mathbf{g})^T)$ defined by the private key. Due to the structural properties of convolutional modular lattices, and the restrictions on the choice of $\mathbf{f}$, the HNF public basis has an especially nice form

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{T}^*\mathbf{h} & q \cdot \mathbf{I} \end{bmatrix} \qquad \text{where } \mathbf{h} = [\mathbf{T}^*\mathbf{f}]^{-1}\mathbf{g} \pmod{q}, \tag{6}$$

  and can be compactly represented just by the vector $\mathbf{h} \in \mathbb{Z}_q^n$.

- **Encryption.** An input message is encoded as a vector $\mathbf{m} \in \{1, 0, -1\}^n$ with exactly $d_f + 1$ positive entries and d_f negative ones. The vector $\mathbf{m}$ is concatenated with a randomly chosen vector $\mathbf{r} \in \{1, 0, -1\}^n$ also with exactly $d_f + 1$ positive entries and d_f negative ones, to obtain a short error vector $(-\mathbf{r}, \mathbf{m}) \in \{1, 0, -1\}^{2n}$. (The multiplication of $\mathbf{r}$ by -1 is clearly unnecessary, and it is performed here just to keep our notation closer to the original description of NTRU. The restriction on the number of nonzero entries is used to bound the probability of decryption errors.) Reducing the error vector $(-\mathbf{r}, \mathbf{m})$ modulo the public basis $\mathbf{H}$ yields

$$\begin{bmatrix} -\mathbf{r} \\ \mathbf{m} \end{bmatrix} \bmod \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{T}^*\mathbf{h} & q \cdot \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ (\mathbf{m} + [\mathbf{T}^*\mathbf{h}]\mathbf{r}) \bmod q \end{bmatrix}.$$

  Since the first n coordinates of this vector are always 0, they can be omitted, leaving only the n-dimensional vector $\mathbf{c} = \mathbf{m} + [\mathbf{T}^*\mathbf{h}]\mathbf{r} \bmod q$ as the ciphertext.

- **Decryption.** The ciphertext $\mathbf{c}$ is decrypted by multiplying it by the secret matrix $[\mathbf{T}^*\mathbf{f}]$ modulo q, yielding

$$[\mathbf{T}^*\mathbf{f}]\mathbf{c} \bmod q = [\mathbf{T}^*\mathbf{f}]\mathbf{m} + [\mathbf{T}^*\mathbf{f}][\mathbf{T}^*\mathbf{h}]\mathbf{r} \bmod q = [\mathbf{T}^*\mathbf{f}]\mathbf{m} + [\mathbf{T}^*\mathbf{g}]\mathbf{r} \bmod q,$$

Estimated Security (bits)	n	q	d_f	key size (bits)
80	257	2^{10}	77	2570
80	449	2^8	24	3592
256	797	2^{10}	84	7970
256	14303	2^8	26	114424

Table 2: Some recommended parameter sets for NTRU public key cryptosystem. Security is expressed in "bits", where k-bits of security roughly means that the best known attack to NTRU requires at least an effort comparable to about 2^k NTRU encryption operations. The parameter d_f is chosen in such a way to ensure the probability of decryption errors (by honest users) is at most 2^{-k}. See [28] for details, and a wider range of parameter choices.

where we have used the identity $[\mathbf{T}^*\mathbf{f}][\mathbf{T}^*\mathbf{h}] = [\mathbf{T}^*([\mathbf{T}^*\mathbf{f}]\mathbf{h})]$ valid for any vectors $\mathbf{f}$ and $\mathbf{h}$. The decryption procedure relies on the fact that the coordinates of the vector

$$[\mathbf{T}^*\mathbf{f}]\mathbf{m} + [\mathbf{T}^*\mathbf{g}]\mathbf{r} \tag{7}$$

are all bounded by $q/2$ in absolute value, so the decrypter can recover the exact value of (7) over the integers (i.e., without reduction modulo q.) The bound on the coordinates of (7) holds provided $d_f < (q/2-1)/(4p) - (1/2)$, or, with high probability, even for larger values of d_f. The decryption process is completed by reducing (7) modulo p, to obtain

$$[\mathbf{T}^*\mathbf{f}]\mathbf{m} + [\mathbf{T}^*\mathbf{g}]\mathbf{r} \bmod p = \mathbf{I}\cdot\mathbf{m} + \mathbf{O}\cdot\mathbf{r} = \mathbf{m}.$$

Algorithm 4 The NTRU public key cryptosystem.

- **Parameters:** Prime n, modulus q, and integer bound d_f. Small integer parameter $p = 3$ is set to a fixed value for simplicity, but other choices are possible.
- **Private key:** Vectors $\mathbf{f} \in \mathbf{e}_1 + \{p, 0, -p\}^n$ and $\mathbf{g} \in \{p, 0, -p\}^n$, such that each of $\mathbf{f} - \mathbf{e}_1$ and $\mathbf{g}$ contains exactly $d_f + 1$ positive entries and d_f negative ones, and the matrix $[\mathbf{T}^*\mathbf{f}]$ is invertible modulo q.
- **Public key:** The vector $\mathbf{h} = [\mathbf{T}^*\mathbf{f}]^{-1}\mathbf{g} \bmod q \in \mathbb{Z}_q^n$.
- **Encryption:** The message is encoded as a vector $\mathbf{m} \in \{1, 0, -1\}^n$, and uses as randomness a vector $\mathbf{r} \in \{1, 0, -1\}^n$, each containing exactly $d_f + 1$ positive entries and d_f negative ones. The encryption function outputs $\mathbf{c} = \mathbf{m} + [\mathbf{T}^*\mathbf{h}]\mathbf{r} \bmod q$.
- **Decryption:** On input ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$, output $(([\mathbf{T}^*\mathbf{f}]\mathbf{c}) \bmod q) \bmod p$, where reduction modulo q and p produces vectors with coordinates in $[-q/2, +q/2]$ and $[-p/2, p/2]$ respectively.

This completes the description of the NTRU cryptosystem, at least for the main set of parameters proposed in [28]. Like GGH, no proof of security supporting NTRU is known, and confidence in the security of the scheme is gained primarily from the best currently known attacks. The strongest attack to NTRU known to date was discovered by Howgrave-Graham [30], who combined previous lattice-based attacks of Coppersmith and Shamir [13], with a combinatorial attack due to Odlyzko (reported in [29, 30, 28]). Based on Howgrave-Graham's hybrid attack, NTRU Cryptosystems issued a collection of recommended parameter sets [28], some of which are reported in Table 2.

5.3 The Ajtai-Dwork cryptosystem and followup work

Following Ajtai's discovery of lattice-based hash functions, Ajtai and Dwork [6] constructed a *public-key cryptosystem* whose security is based on the worst-case hardness of a lattice problem. Several improvements

were given in subsequent works [21, 69], mostly in terms of the security proof and simplifications to the cryptosystem. In particular, the cryptosystem in [69] is quite simple as it only involves modular operations on integers, though much longer ones than those typically used in lattice-based cryptography.

Unlike the case of hash functions, the security of these cryptosystems is based on the worst-case hardness of a special case of SVP known as unique-SVP. Here, we are given a lattice whose shortest nonzero vector is shorter by some factor γ than all other nonparallel lattice vectors, and our goal is to find a shortest nonzero lattice vector. The hardness of this problem is not understood as well as that of SVP, and it is a very interesting open question whether one can base public-key cryptosystems on the (worst-case) hardness of SVP.

The aforementioned lattice-based cryptosystems are unfortunately quite inefficient. It turns out that when we base the security on lattices of dimension n, the size of the public key is $\tilde{O}(n^4)$ and each encrypted bit gets blown up to $\tilde{O}(n^2)$ bits. So if, for instance, we choose n to be several hundreds, the public key size is on the order of several gigabytes, which clearly makes the cryptosystem impractical.

Ajtai [5] also presented a more efficient cryptosystem whose public key scales like $\tilde{O}(n^2)$ and in which each encrypted bit gets blown up to $\tilde{O}(n)$ bits. The size of the public key can be further reduced to $\tilde{O}(n)$ if one can set up a pre-agreed trusted random string of length $\tilde{O}(n^2)$. Unfortunately, the security of this cryptosystem is not known to be as strong as that of other lattice-based cryptosystems: it is based on a problem by Dirichlet, which is not directly related to any standard lattice problem. Moreover, this system has no worst-case hardness as the ones previously mentioned. Nevertheless, the system does have the flavor of a lattice-based cryptosystem.

5.4 The LWE-based cryptosystem

In this section we describe what is perhaps the most efficient lattice-based cryptosystem to date supported by a theoretical proof of security. The first version of the cryptosystem together with a security proof were presented by Regev [71]. Some improvements in efficiency were suggested by Kawachi et al. [32]. Then, some very significant improvements in efficiency were given by Peikert et al. [65]. The cryptosystem we describe here is identical to the one in [65] except for one additional optimization that we introduce (namely, the parameter r). Another new optimization based on the use of the Hermite Normal Form [47] is described separately at the end of the subsection. When based on the hardness of lattice problems in dimension n, the cryptosystem has a public key of size $\tilde{O}(n^2)$, requires $\tilde{O}(n)$ bit operations per encrypted bit, and expands each encrypted bit to $O(1)$ bits. This is considerably better than those proposals following the Ajtai-Dwork construction, but is still not ideal, especially in terms of the public key size. We will discuss these issues in more detail later, as well as the possibility of reducing the public key size by using restricted classes of lattices such as cyclic lattices.

The cryptosystem was shown to be secure (under chosen plaintext attacks) based on the conjectured hardness of the *learning with errors* problem (LWE), which we define next. This problem is parameterized by integers n, m, q and a probability distribution χ on $\mathbb{Z}_q$, typically taken to be a "rounded" normal distribution. The input is a pair $(\mathbf{A}, \mathbf{v})$ where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is chosen uniformly, and $\mathbf{v}$ is either chosen uniformly from $\mathbb{Z}_q^m$ or chosen to be $\mathbf{As} + \mathbf{e}$ for a uniformly chosen $\mathbf{s} \in \mathbb{Z}_q^n$ and a vector $\mathbf{e} \in \mathbb{Z}_q^m$ chosen according to χ^m. The goal is to distinguish with some non-negligible probability between these two cases. This problem can be equivalently described as a bounded distance decoding problem in q-ary lattices: given a uniform $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{v} \in \mathbb{Z}_q^m$ we need to distinguish between the case that $\mathbf{v}$ is chosen uniformly from $\mathbb{Z}_q^m$ and the case in which $\mathbf{v}$ is chosen by perturbing each coordinate of a random point in $\Lambda_q(\mathbf{A}^T)$ using χ.

The LWE problem is believed to be very hard (for reasonable choices of parameters), with the best known algorithms running in exponential time in n (see [71]). Several other facts lend credence to the conjectured hardness of LWE. First, the LWE problem can be seen as an extension of a well-known problem in learning theory, known as the *learning parity with noise* problem, which in itself is believed to be very hard. Second, LWE is closely related to decoding problems in coding theory which are also believed to be very hard. Finally, the LWE was shown to have a worst-case connection, as will be discussed below. In Section 7 we will present several other cryptographic constructions based on the LWE problem.

The worst-case connection: A reduction from worst-case lattice problems such as approximate-SVP and approximate-SIVP to LWE was established in [71], giving a strong indication that the LWE problem is hard. This reduction, however, is a *quantum* reduction, i.e., the algorithm performing the reduction is a quantum algorithm. What this means is that hardness of LWE (and hence the security of the cryptosystem) is established based on the worst-case *quantum* hardness of approximate-SVP. In other words, breaking the cryptosystem (or finding an efficient algorithm for LWE) implies an efficient quantum algorithm for approximating SVP, which, as discussed in Subsection 1.3, would be very surprising. This security guarantee is incomparable to the one by Ajtai and Dwork: On one hand, it is stronger as it is based on the general SVP and not the special case of unique-SVP. On the other hand, it is weaker as it only implies a *quantum* algorithm for lattice problems.

The reduction is described in detail in the following theorem, whose proof forms the main bulk of [71]. For a real $\alpha > 0$ we let $\bar{\Psi}_\alpha$ denote the distribution on $\mathbb{Z}_q$ obtained by sampling a normal variable with mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$, rounding the result to the nearest integer and reducing it modulo q.

Theorem 5.1 ([71]) *Assume we have access to an oracle that solves the* LWE *problem with parameters* $n, m, q, \bar{\Psi}_\alpha$ *where* $\alpha q > \sqrt{n}$, $q \leq \text{poly}(n)$ *is prime, and* $m \leq \text{poly}(n)$. *Then there exists a quantum algorithm running in time* $\text{poly}(n)$ *for solving the (worst-case) lattice problems* $\mathsf{SIVP}_{\tilde{O}(n/\alpha)}$ *and (the decision variant of)* $\mathsf{SVP}_{\tilde{O}(n/\alpha)}$ *in* any *lattice of dimension* n.

Notice that m plays almost no role in this reduction and can be taken to be as large as one wishes (it is not difficult to see that the problem can only become easier for larger m). It is possible that this reduction to LWE will one day be "dequantized" (i.e., made non-quantum), leading to a stronger security guarantee for LWE-based cryptosystems. Finally, let us emphasize that quantum arguments show up only in the reduction to LWE — the LWE problem itself, as well as all cryptosystems based on it are entirely classical.

The cryptosystem: The cryptosystem is given in Algorithm 5, and is partly illustrated in Figure 3. It is parameterized by integers n, m, ℓ, t, r, q, and a real $\alpha > 0$. The parameter n is in some sense the main security parameter, and it corresponds to the dimension of the lattices that show up in the worst-case connection. We will later discuss how to choose all other parameters in order to guarantee security and efficiency. The message space is $\mathbb{Z}_t^\ell$. We let f be the function that maps the message space $\mathbb{Z}_t^\ell$ to $\mathbb{Z}_q^\ell$ by multiplying each coordinate by q/t and rounding to the nearest integer. We also define an "inverse" mapping f^{-1} which takes an element of $\mathbb{Z}_q^\ell$ and outputs the element of $\mathbb{Z}_t^\ell$ obtained by dividing each coordinate by q/t and rounding to the nearest integer.

Algorithm 5 The LWE-based public key cryptosystem.

- **Parameters:** Integers n, m, ℓ, t, r, q, and a real $\alpha > 0$.
- **Private key:** Choose $\mathbf{S} \in \mathbb{Z}_q^{n \times \ell}$ uniformly at random. The private key is $\mathbf{S}$.
- **Public key:** Choose $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ uniformly at random and $\mathbf{E} \in \mathbb{Z}_q^{m \times \ell}$ by choosing each entry according to $\bar{\Psi}_\alpha$. The public key is $(\mathbf{A}, \mathbf{P} = \mathbf{AS} + \mathbf{E}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times \ell}$.
- **Encryption:** Given an element of the message space $\mathbf{v} \in \mathbb{Z}_t^\ell$, and a public key $(\mathbf{A}, \mathbf{P})$, choose a vector $\mathbf{a} \in \{-r, -r+1, \ldots, r\}^m$ uniformly at random, and output the ciphertext $(\mathbf{u} = \mathbf{A}^T\mathbf{a}, \mathbf{c} = \mathbf{P}^T\mathbf{a} + f(\mathbf{v})) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.
- **Decryption:** Given a ciphertext $(\mathbf{u}, \mathbf{c}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$ and a private key $\mathbf{S} \in \mathbb{Z}_q^{n \times \ell}$, output $f^{-1}(\mathbf{c} - \mathbf{S}^T\mathbf{u})$.

5.4.1 Choosing the parameters

The choice of parameters is meant to guarantee efficiency, a low probability of decryption errors, and security. We now discuss these issues in detail.

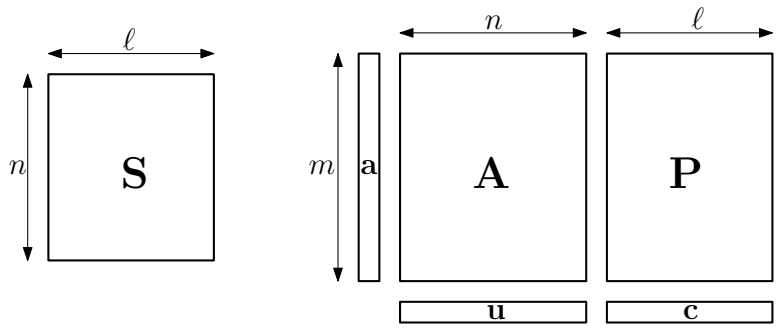

Figure 3: Ingredients in the LWE-based cryptosystem.

Efficiency: The cryptosystem is clearly very easy to implement, as it involves nothing but additions and multiplications modulo q. Some improvement in running time can be obtained by setting t to be a power two (which simplifies the task of converting an input message into an element of the message space), and by postponing the modular reduction operations (assuming, of course, that registers are large enough so that no overflow occurs). Moreover, high levels of parallelization are easy to obtain.

In the following we list some properties of the cryptosystem, all of which are easy to observe. All sizes are in bits, logarithms are base 2, and the $\tilde{O}(\cdot)$ notation hides logarithmic factors.

- Private key size: $n\ell \log q$
- Public key size: $m(n+\ell)\log q$
- Message size: $\ell \log t$
- Ciphertext size: $(n+\ell)\log q$
- Encryption blowup factor: $(1+\frac{n}{\ell})\log q/\log t$
- Operations for encryption per bit: $\tilde{O}(m(1+\frac{n}{\ell}))$
- Operations for decryption per bit: $\tilde{O}(n)$

Decryption errors: The cryptosystem has some positive probability of decryption errors. This probability can be made very small with an appropriate setting of parameters. Moreover, if an error correcting code is used to encode the messages before encryption, this error probability can be reduced to undetectable levels.

We now estimate the probability of a decryption error in one letter, i.e., an element of $\mathbb{Z}_t$ (recall that each message consists of ℓ letters). Assume we choose a private key $\mathbf{S}$, public key $(\mathbf{A}, \mathbf{P})$, encrypt some message $\mathbf{v}$ and then decrypt it. The result is given by

$$\begin{aligned} f^{-1}(\mathbf{c} - \mathbf{S}^T\mathbf{u}) &= f^{-1}(\mathbf{P}^T\mathbf{a} + f(\mathbf{v}) - \mathbf{S}^T\mathbf{A}^T\mathbf{a}) \\ &= f^{-1}((\mathbf{AS}+\mathbf{E})^T\mathbf{a} + f(\mathbf{v}) - \mathbf{S}^T\mathbf{A}^T\mathbf{a}) \\ &= f^{-1}(\mathbf{E}^T\mathbf{a} + f(\mathbf{v})). \end{aligned}$$

Hence, in order for a letter decryption error to occur, say in the first letter, the first coordinate of $\mathbf{E}^T\mathbf{a}$ must be greater than $q/(2t)$ in absolute value. Fixing the vector $\mathbf{a}$ and ignoring the rounding, the distribution of the first coordinate of $\mathbf{E}^T\mathbf{a}$ is a normal distribution with mean 0 and standard deviation $\alpha q\|\mathbf{a}\|/\sqrt{2\pi}$ since the sum of independent normal variables is still a normal variable with the variance begin the sum of variances. Now the norm of $\mathbf{a}$ can be seen to be with very high probability close to

$$\|\mathbf{a}\| \approx \sqrt{r(r+1)m/3}.$$

To see this, recall that each coordinate of $\mathbf{a}$ is distributed uniformly on $\{-r, \ldots, r\}$. Hence, the expectation squared of each coordinate is

$$\frac{1}{2r+1} \sum_{k=-r}^{r} k^2 = \frac{r(r+1)}{3}$$

from which it follows that $\|\mathbf{a}\|^2$ is tightly concentrated around $r(r+1)m/3$.

The error probability per letter can now be estimated by the probability that a normal variable with mean 0 and standard deviation $\alpha q \sqrt{r(r+1)m/(6\pi)}$ is greater in absolute value than $q/(2t)$, or equivalently,

$$\text{error probability per letter} \;\approx\; 2\left(1 - \Phi\left(\frac{1}{2t\alpha} \cdot \sqrt{\frac{6\pi}{r(r+1)m}}\right)\right) \tag{8}$$

where Φ here is the cumulative distribution function of the standard normal distribution. For most reasonable choices of parameters, this estimate is in fact very close to the true error probability.

Security: The proof of security, as given in [71] and [65], consists of two main parts. In the first part, one shows that distinguishing between public keys $(\mathbf{A}, \mathbf{P})$ as generated by the cryptosystem and pairs $(\mathbf{A}, \mathbf{P})$ chosen uniformly at random from $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times \ell}$ implies a solution to the LWE problem with parameters $n, m, q, \bar{\Psi}_\alpha$. Hence if we set n, m, q, α to values for which we believe LWE is hard, we obtain that the public keys generated by the cryptosystem are indistinguishable from pairs chosen uniformly at random. The second part consists of showing that if one tries to encrypt with a public key $(\mathbf{A}, \mathbf{P})$ chosen at random, then with very high probability, the result carries essentially no statistical information about the encrypted message (this is what [65] refer to as "messy keys"). Together, these two parts establish the security of the cryptosystem (under chosen plaintext attacks). The argument is roughly the following: due to the second part, being able to break to system, even with some small non-negligible probability, implies the ability to distinguish valid public keys from uniform pairs, but this task is hard due to the first part.

In order to guarantee security, our choice of parameters has to be such that the two properties above are satisfied. Let us start with the second one. Our goal is to guarantee that when $(\mathbf{A}, \mathbf{P})$ is chosen uniformly, the encryptions carry no information about the message. For this, it would suffice to guarantee that $(\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$ is essentially uniformly distributed (since in this case the shift by $f(\mathbf{v})$ is essentially unnoticeable). By following an argument similar to the one in [71, 65], one can show that a sufficient condition for this is that the number of possibilities for $\mathbf{a}$ is much larger than the number of elements in our range, i.e.,

$$(2r+1)^m \gg q^{n+\ell}. \tag{9}$$

More precisely, the statistical distance from the uniform distribution is upper bounded by the square root of the ratio between the two quantities, and hence the latter should be negligible, say 2^{-100}.

We now turn to the first property. Our goal is to choose n, m, q, α so that the LWE problem is hard. One guiding principle we can use is the worst-case connection, as described in Theorem 5.1. This suggest that the choice of m is inconsequential, that q should be prime, that αq should be bigger than $\sqrt{n}$, and that α should be as big as possible (as it leads to harder worst-case problems). Unfortunately, the worst-case connection does not seem to provide hints on actual security for any concrete choice of parameters. For this, one has to take into account experiments on the hardness of LWE, as we discuss next.

In order to estimate the hardness of LWE for a concrete set of parameters, recall that the LWE can be seen as a certain bounded distance decoding problem on q-ary lattices. Namely, we are given a point $\mathbf{v}$ that is either close to $\Lambda_q(\mathbf{A}^T)$ (with the perturbation in each coordinate chosen according to $\bar{\Psi}_\alpha$) or uniform. One natural approach to try to distinguish between these two cases is to find a short vector $\mathbf{w}$ in the dual lattice $\Lambda_q(\mathbf{A}^T)^*$ and check the inner product $\langle \mathbf{v}, \mathbf{w} \rangle$: if $\mathbf{v}$ is close to the lattice, this inner product will tend to be close to an integer. This method is effective as long as the perturbation in the direction of $\mathbf{w}$ is not much bigger than $1/\|\mathbf{w}\|$. Since our perturbation is (essentially) Gaussian, its standard deviation in any direction

(and in particular in the direction of $\mathbf{w}$) is $\alpha q/\sqrt{2\pi}$. Therefore, in order to guarantee security, we need to ensure that

$$\alpha q/\sqrt{2\pi} \gg 1/\|\mathbf{w}\|.$$

A factor of 1.5 between the two sides of the inequality is sufficient to guarantee that the observed distribution of $\langle \mathbf{v}, \mathbf{w} \rangle \bmod 1$ is within negligible statistical distance of uniform.

Using the results of Section 3, we can predict that the shortest vector found by the best known lattice reduction algorithms when applied to the lattice $\Lambda_q(\mathbf{A}^T)^* = \frac{1}{q}\Lambda_q^\perp(\mathbf{A}^T)$ is of length

$$\|\mathbf{w}\| \approx \frac{1}{q} \cdot \min\{q, 2^{2\sqrt{n \log q \log \delta}}\}$$

and that in order to arrive at such a vector (assuming the minimum is achieved by the second term) one needs to apply lattice reduction to lattices of dimension

$$\sqrt{n \log q / \log \delta}. \tag{10}$$

We therefore obtain the requirement

$$\alpha \geq 1.5\sqrt{2\pi} \max\left\{\frac{1}{q}, 2^{-2\sqrt{n \log q \log \delta}}\right\}. \tag{11}$$

The parameter m again seems to play only a minor role in the practical security of the system.

Choice of parameters: By taking the above discussion into account, we can now finally give some concrete choices of parameters that seem to guarantee both security and efficiency. To recall, the system has seven parameters, n, ℓ, q, r, t, m and α. In order to guarantee security, we need to satisfy Eqs. (9) and (11). To obtain the former, we set

$$m = ((n + \ell) \log q + 200)/\log(2r + 1).$$

Next, following Eq. (11), we set

$$\alpha = 4 \cdot \max\left\{\frac{1}{q}, 2^{-2\sqrt{n \log q \log(1.01)}}\right\}.$$

Our choice of $\delta = 1.01$ seems reasonable for the lattice dimensions with which we are dealing here; one can also consider more conservative choices like $\delta = 1.005$.

We are thus left with five parameters, n, ℓ, q, r, and t. We will choose them in an attempt to optimize the following measures.

- Public key size: $m(n + \ell) \log q$
- Encryption blowup factor: $(1 + \frac{n}{\ell}) \log q / \log t$
- Error probability per letter:
$$2\left(1 - \Phi\left(\frac{1}{2t\alpha} \cdot \sqrt{\frac{6\pi}{r(r+1)m}}\right)\right)$$
- Lattice dimension involved in best known attack: $\sqrt{n \log q / \log(1.01)}$

As a next step, notice that ℓ should not be much smaller than n as this makes the encryption blowup factor very large. For concreteness we choose $\ell = n$, which gives a fair balance between the encryption blowup factor and the public key size. Denoting $N = n \log q$, we are thus left with the following measures.

- Public key size: $2N(2N + 200)/\log(2r + 1)$

- Encryption blowup factor: $2\log q/\log t$
- Error probability per letter:

$$2\left(1-\Phi\left(\frac{1}{8t}\min\{q,2^{2\sqrt{N\log(1.01)}}\}\cdot\sqrt{\frac{6\pi}{r(r+1)(2N+200)/\log(2r+1)}}\right)\right)$$

- Lattice dimension involved in best known attack: $\sqrt{N/\log(1.01)}$

Finally, once we fix $N = n\log q$, we should choose q as small as possible and r and t as large as possible while still keeping the error probability within the desired range.

Some examples are given in Table 3. In all examples we took $\ell = n$, and tried to minimize either the public key size or the encryption blowup factor while keeping the error probability below 1%. To recall, this error probability can be made negligible by using an error correcting code. The public key size can be decreased by up to a factor of 2 by choosing a smaller ℓ (at the expense of higher encryption blowup).

n	136	166	192	214	233	233
ℓ	136	166	192	214	233	233
m	2008	1319	1500	1333	1042	4536
q	2003	4093	8191	16381	32749	32749
r	1	4	5	12	59	1
t	2	2	4	4	2	40
α	0.0065	0.0024	0.0009959	0.00045	0.000217	0.000217
Public key size in bits	6×10^6	5.25×10^6	7.5×10^6	8×10^6	7.3×10^6	31.7×10^6
Encryption blowup factor	21.9	24	13	14	30	5.6
Error probability	0.9%	0.56%	1%	0.8%	0.9%	0.9%
Lattice dimension in attack	322	372	417	457	493	493

Table 3: Some possible choices of parameters using $\delta = 1.01$.

Further optimizations: If all users of the system have access to a trusted source of random bits, they can use it to agree on a random matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$. This allows us to include only $\mathbf{P}$ in the public key, thereby reducing its size to $m\ell\log q$, which is $\tilde{O}(n)$ if ℓ is chosen to be constant and $m = \tilde{O}(n)$. This observation, originally due to Ajtai [5], crucially relies on the source of random bits being trusted, since otherwise it might contain a trapdoor (see [17]). Moreover, as already observed, choosing small ℓ results in large ciphertext blowup factors. If ℓ is set to $O(n)$ in order to achieve constant encryption blowup, then the public key will have size at least $\tilde{O}(n^2)$ even if a common random matrix is used for $\mathbf{A}$.

Another possible optimization results from the HNF technique of [47] already discussed in the context of the GGH cryptosystem. The improvement it gives is quantitatively modest: it allows to shrink the public key size and encryption times by a factor of $(1 - n/m)$. Still, the improvement comes at absolutely no cost, so it seems well worth adopting in any implementation of the system. Recall that the public key consists of a public lattice $\Lambda_q(\mathbf{A}^T)$ represented by a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, and a collection $\mathbf{AS} + \mathbf{E} \bmod q$ of perturbed lattice vectors $\mathbf{As}_i \in \Lambda_q(\mathbf{A}^T)$. As in the HNF modification of the GGH cryptosystem, cryptanalysis only gets harder if we describe the public lattice by its lower triangular HNF basis, and the perturbed lattice vectors are replaced by the result of reducing the error vectors (i.e., the columns of $\mathbf{E}$) by such a basis.

In more detail, let $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ be chosen uniformly as before. For simplicity, assume $\mathbf{A}$ has full rank (which happens with probability exponentially close to 1), and that its first n rows are linearly independent over $\mathbb{Z}_q$ (which can be obtained by permuting its rows). Under these conditions, the q-ary lattice $\Lambda_q(\mathbf{A}^T)$ has a very simple HNF basis of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{A}' & q\mathbf{I} \end{bmatrix}$$

where $\mathbf{A}' \in \mathbb{Z}_q^{(m-n)\times n}$. Let $\mathbf{E}$ be an error matrix chosen as before, and write it as $\mathbf{E} = (\mathbf{E}'', \mathbf{E}')$ where $\mathbf{E}'' \in \mathbb{Z}_q^{n\times \ell}$ and $\mathbf{E}' \in \mathbb{Z}_q^{(m-n)\times \ell}$. Reducing the columns of $\mathbf{E}$ modulo the HNF public basis $\mathbf{H}$ yields vectors $(\mathbf{O}, \mathbf{P}')$ where $\mathbf{P}' = \mathbf{E}' - \mathbf{A}'\mathbf{E}'' \in \mathbb{Z}_q^{(m-n)\times \ell}$. The public key consists of $(\mathbf{I}, \mathbf{A}') \in \mathbb{Z}_q^{m\times n}$ and $(\mathbf{O}, \mathbf{P}') \in \mathbb{Z}_q^{m\times \ell}$. Since $\mathbf{I}$ and $\mathbf{O}$ are fixed matrices, only $\mathbf{A}'$ and $\mathbf{P}'$ need to be stored as part of the public key, reducing the public key bit-size to $(m-n)(n+\ell)\log q$. Encryption proceeds as before, i.e., the ciphertext is given by

$$(\mathbf{u}, \mathbf{c}) = (\mathbf{a}'' + (\mathbf{A}')^T\mathbf{a}', (\mathbf{P}')^T\mathbf{a}' + f(\mathbf{v}))$$

where $\mathbf{a} = (\mathbf{a}'', \mathbf{a}')$. Notice that the secret matrix $\mathbf{S}$ used by the original LWE cryptosystem has disappeared. The matrix $\mathbf{E}'' \in \mathbb{Z}_q^{n\times \ell}$ is used instead for decryption. Given ciphertext $(\mathbf{u}, \mathbf{c})$, the decrypter outputs $f^{-1}(\mathbf{c} + (\mathbf{E}'')^T\mathbf{u})$. Notice that the vector $\mathbf{c} + (\mathbf{E}'')^T\mathbf{u}$ still equals $(\mathbf{E}^T\mathbf{a}) + f(\mathbf{v})$, so decryption will succeed with exactly the same probability as the original LWE cryptosystem. The security of the system can be established by a reduction from the security of the original cryptosystem. To conclude, this modification allows us to shrink the public key size and encryption time by a factor of $(1 - n/m)$ at no cost.

6 Digital Signature Schemes

Digital signature schemes are among the most important cryptographic primitives. From a theoretical point of view, signature schemes can be constructed from one-way functions in a black-box way without any further assumptions [56]. Therefore, by using the one-way functions described in Section 4 we can obtain signature schemes based on the worst-case hardness of lattice problems. These black-box constructions, however, incur a large overhead and are impractical. In this section we survey some proposals for signature schemes that are directly based on lattice problems, and are typically much more efficient.

The earliest proposal for a lattice-based signature scheme was given by Goldreich et al. [22], and is based on ideas similar to those in their cryptosystem described in Subsection 5.1. In 2003, the company NTRU Cryptosystems proposed an efficient signature scheme called NTRUSign [26]. This signature scheme can be seen as an optimized instantiation of the GGH scheme, based on the NTRU lattices. Unfortunately, both schemes (in their basic version) can be broken in a strong asymptotic sense. We remark that neither scheme came with a security proof, which explains the serious security flaws which we will describe later.

The first construction of efficient signature schemes with a supporting proof of security (in the random oracle model) was suggested by Micciancio and Vadhan [55], who gave statistical zero knowledge proof systems for various lattice problems, and observed that such proof systems can be converted in a relatively efficient way first into secure identification schemes, and then (via the Fiat-Shamir heuristic) into a signature scheme in the random oracle model. More efficient schemes were recently proposed by Lyubashevsky and Micciancio [44], and by Gentry, Peikert and Vaikuntanathan [17]. Interestingly, the latter scheme can be seen as a theoretically justified variant of the GGH and NTRUSign signature schemes, with worst-case security guarantees based on general lattices in the random oracle model. The scheme of Lyubashevsky and Micciancio [44] has worst-case security guarantees based on ideal lattices similar to those considered in the construction of hash functions (see Section 4), and it is the most (asymptotically) efficient construction known to date, yielding signature generation and verification algorithms that run in almost linear time. Moreover, the security of [44] does not rely on the random oracle model.

In the rest of this section we describe the GGH and NTRUSign signature schemes, and the security flaw in their design, the theoretically justified variant of their scheme proposed by Gentry et al., and finally the signature scheme of Lyubashevsky and Micciancio, which is currently the most efficient (lattice-based) signature scheme with a supporting proof of security, at least in an asymptotic sense.

Lattice-based digital signature schemes have not yet reached the same level of maturity as the collision resistant hash functions and public key encryption schemes presented in the previous sections. So, in this section we present the schemes only informally, and refer the reader to the original papers (and any relevant literature appearing after the time of this writing) for details.

6.1 The GGH and NTRUSign signature schemes

We now briefly describe the GGH signature scheme; for a description of NTRUSign, see [26]. The private and public keys are chosen as in the GGH encryption scheme. That is, the private key is a lattice basis $\mathbf{B}$ consisting of short and fairly orthogonal vectors. The public key $\mathbf{H}$ is a "bad" basis for the same lattice $\mathcal{L}(\mathbf{B})$, i.e., a basis consisting of fairly long and far from orthogonal vectors. As before, it is best to choose $\mathbf{H}$ to be the Hermite normal form of $\mathbf{B}$.

To sign a given message, we first map it to a point $\mathbf{m} \in \mathbb{R}^n$ using some hash function. We assume that the hash function behaves like a random oracle, so that $\mathbf{m}$ is distributed uniformly (in some large volume of space). Next, we round $\mathbf{m}$ to a nearby lattice point $\mathbf{s} \in \mathcal{L}(\mathbf{B})$ by using the secret basis. This is typically done using Babai's round-off procedure [8], which gives

$$\mathbf{s} = \mathbf{B}\lfloor \mathbf{B}^{-1}\mathbf{m} \rceil.$$

Notice that by definition, this implies that

$$\mathbf{s} - \mathbf{m} \in \mathcal{P}_{1/2}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in [-1/2, 1/2]^n\}.$$

In order to verify a given message-signature pair $(\mathbf{m}, \mathbf{s})$, one checks that $\mathbf{s} \in \mathcal{L}(\mathbf{H}) = \mathcal{L}(\mathbf{B})$ (which can be done efficiently using the public key $\mathbf{H}$) and that the distance $\|\mathbf{s} - \mathbf{m}\|$ is small (which should be the case since this difference is contained in $\mathcal{P}_{1/2}(\mathbf{B})$).

Attacks: Some early indications that the GGH and NTRUSign signature schemes might be insecure were given by Gentry and Szydlo [18, 77] who observed that each signature leaks some information on the secret key. This information leakage does not necessarily prove that such schemes are insecure, since it might be computationally difficult to use this information. However, as was demonstrated by Nguyen and Regev a few years later [59], this information leakage *does* lead to an attack on the scheme. More precisely, they have shown that given enough message-signature pairs, it is possible to recover the private key. Moreover, their attack is quite efficient, and was implemented and applied in [59] to most reasonable choices of parameters in GGH and NTRUSign, thereby establishing that these signature schemes are not secure in practice (but see below for the use of "perturbations" in NTRUSign).

The idea behind the information leakage and the attack is in fact quite simple. The basic observation is that the difference $\mathbf{m} - \mathbf{s}$ obtained from a message-signature pair $(\mathbf{m}, \mathbf{s})$ is distributed essentially uniformly in $\mathcal{P}_{1/2}(\mathbf{B})$. Hence, given enough such pairs, we end up with the following algorithmic problem, called the hidden parallelepiped problem (see Fig. 4): given many random points uniformly distributed over an unknown n-dimensional parallelepiped, recover the parallelepiped or an approximation thereof. An efficient solution to this problem implies the attack mentioned above.

In the two-dimensional case shown in Fig. 4, one immediately *sees* the parallelepiped enveloping the points, and it is not difficult to come up with an algorithm that implements this. But what about the high-dimensional case? High dimensional problems are often very hard. Here, however, the problem turns out to be easy. The algorithm used in [59] applies a gradient decent method to solve a multivariate optimization problem based on the fourth-moment of the one-dimensional projections. See [59] for further details (as well as for an interesting historical account of the hidden parallelepiped problem).

Countermeasures: The most efficient countermeasures known against the above attack are perturbation techniques [26, 27]. These modify the signature generation process in such a way that the hidden parallelepiped is replaced by a considerably more complicated body, and this seems to prevent attacks of the type described above. The main drawback of perturbations is that they slow down signature generation and increase the size of the secret key. Nevertheless, the NTRUSign signature scheme with perturbation is still relatively efficient. Finally, notice that even with perturbations, NTRUSign does not have any security proof.

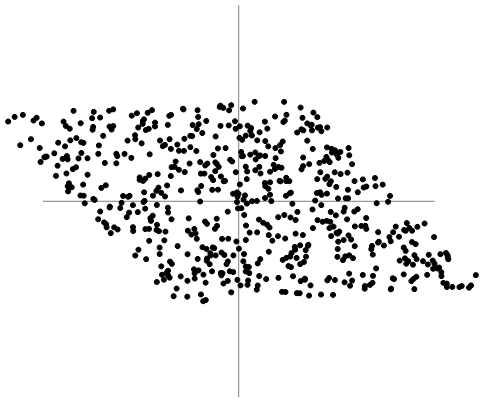

Figure 4: The hidden parallelepiped problem in two dimensions.

6.2 Schemes based on preimage sampleable trapdoor functions

In a recent paper, Gentry, Peikert, and Vaikuntanathan [17] defined an abstraction called "preimage sampleable trapdoor functions", and showed how to instantiate it based on the worst-case hardness of lattice problems. They then showed that this abstraction is quite powerful: it can be used instead of trapdoor permutations in several known constructions of signature schemes in the random oracle model. This leads to relatively efficient signature schemes that are provably secure (in the random oracle model) based on the worst-case hardness of lattice problems.

One particularly interesting feature of their construction is that it can be seen as a provably secure variant of the (insecure) GGH scheme. Compared to the GGH scheme, their construction differs in two main aspects. First, it is based on lattices chosen from a distribution that enjoys a worst-case connection (the lattices in GGH and NTRU are believed to be hard, but not known to have a worst-case connection). A second and crucial difference is that their signing algorithm is designed so that it does not reveal any information about the secret basis. This is achieved by replacing Babai's round-off procedure with a "Gaussian sampling procedure", originally due to Klein [35], whose distinctive feature is that its output distribution, for the range of parameters considered in [17], is essentially independent of the secret basis used. The effect of this on the attack outlined above is that instead of observing points chosen uniformly from the parallelepiped generated by the secret basis, the attack observes points chosen from a spherically symmetric Gaussian distribution, and therefore learns nothing about the secret basis. The Gaussian sampling procedure is quite useful, and has already lead to the development of several other lattice-based constructions, as will be mentioned in Section 7.

As most schemes based on general lattices, the signatures of [17] have quadratic complexity both in terms of key size and signing and verification times. It should be remarked that although most of the techniques from [17] apply to any lattice, it is not clear how to obtain substantially more efficient instantiations of their signatures using structured lattices (e.g., NTRU lattices, or the cyclic/ideal lattices used in the construction of hash functions). For example, even when instantiated with NTRU lattices, the running time of the signing algorithm seems to remain quadratic in the security parameter because of the expensive sampling procedure.

6.3 Schemes based on collision resistant hash functions

Finally, in [44], Lyubashevsky and Micciancio gave a signature scheme which is seemingly optimal on all fronts, at least asymptotically: it admits a proof of security based on worst-case complexity assumptions, the proof of security holds in the standard computational model (no need for random oracles), and the scheme is asymptotically efficient, with key size and signing/verification times all almost linear in the dimension of the underlying lattice. The lattice assumption underlying this scheme is that no algorithm can approximate SVP to within polynomial factors in all ideal lattices, i.e., lattices that are closed under some linear transformation

F of the kind considered in Section 4.

The scheme makes use of a new hash-based one-time signature scheme, i.e., a signature scheme that allows to securely sign a single message. Such schemes can be transformed into full-fledged signature schemes using standard tree constructions (dating back to [24, 57]), with only a logarithmic loss in efficiency. The one-time signature scheme, in turn, is based on a collision resistant hash function based on ideal lattices, of the kind discussed in Section 4. The hash function h can be selected during the key generation process, or be a fixed global parameter. The assumption is that finding collisions in h is computationally hard. The input to h can be interpreted as a sequence of vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{m/n} \in \mathbb{Z}_q^n$ with small coordinates. The secret key to the hash function is a pair of randomly chosen inputs $\mathbf{x}_1, \ldots, \mathbf{x}_{m/n} \in \mathbb{Z}_q^n$ and $\mathbf{y}_1, \ldots, \mathbf{y}_{m/n} \in \mathbb{Z}_q^n$, each chosen according to an appropriate distribution that generates short vectors with high probability.[3] The public key is given by the images of these two inputs under the hash function $X = h(\mathbf{x}_1, \ldots, \mathbf{x}_{m/n}), Y = h(\mathbf{y}_1, \ldots, \mathbf{y}_{m/n})$. Messages to be signed are represented by short vectors $\mathbf{m} \in \mathbb{Z}_q^n$. The signature of a message $\mathbf{m}$ is simply computed as

$$\sigma = (\sigma_1, \ldots, \sigma_{m/n}) = ([\mathbf{F}^*\mathbf{m}]\mathbf{x}_1 + \mathbf{y}_1, \ldots, [\mathbf{F}^*\mathbf{m}]\mathbf{x}_{m/n} + \mathbf{y}_{m/n}) \bmod q.$$

The signature is verified by checking that σ is a sequence of short vectors that hashes to $[\mathbf{F}^*\mathbf{m}]X + Y \bmod q$.

The security of the scheme relies on the fact that even after seeing a signature, the exact value of the secret key is still information theoretically concealed from the adversary. Therefore, if the adversary manages to come up with a forged signature, it is likely to be different from the one that the legitimate signer can compute using the secret key. Since the forged signature and legitimate signature hash to the same value, they provide a collision in the hash function.

7 Other Cryptographic Primitives

In this section we briefly survey lattice-based constructions of other cryptographic primitives. Previous constructions of these primitives were based on (sometimes non-standard) number theoretic assumptions. Since all these constructions are very recent, we will not provide too many details.

CCA-secure cryptosystems: All the cryptosystems mentioned in Section 5 are secure only under chosen plaintext attacks (CPA), and not under chosen ciphertext attacks (CCA). Indeed, it is not difficult to see that given access to the decryption oracle, one can recover the private key. For certain applications, security against CCA attacks is necessary.

CCA-secure cryptosystems are typically constructed based on specific number theoretic assumptions (or in the random oracle model) and no general constructions in the standard model were known till very recently. In a recent breakthrough, Peikert and Waters [66] showed for the first time how to construct CCA-secure cryptosystems based on a general primitive which they call *lossy trapdoor functions*. They also showed how to construct this primitive based either on traditional number theoretic assumptions or on the LWE problem. The latter result is particularly important as it gives for the first time a CCA-secure cryptosystem based on the worst-case (quantum) hardness of lattice problems.

IBE: Gentry et al. [17] have recently constructed identity based encryption (IBE) schemes based on LWE. Generally speaking, IBE schemes are difficult to construct and only a few other proposals are known; the fact that IBE schemes can be based on the LWE problem (and hence on the worst-case quantum hardness of lattice problems) is therefore quite remarkable.

OT protocols: In another recent work, Peikert, Vaikuntanathan, and Waters [65] provide a construction of an oblivious transfer (OT) protocol that is both universally composable and relatively efficient. Their construction can be based on a variety of cryptographic assumptions, and in particular on the LWE problem

[3] For technical reasons, the input vectors cannot be chosen simply uniformly at random from a set of short vectors without invalidating the proof.

(and hence on the worst-case quantum hardness of lattice problems). Such protocols are often used in secure multiparty computation.

Zero-Knowledge proofs and ID schemes: Various zero-knowledge proof systems and identification schemes were recently discovered. *Interactive* statistical zero-knowledge proof systems for various lattice problems (including approximate SVP) were already given by Micciancio and Vadhan in [55]. In [64], Peikert and Vaikuntanathan gave *non-interactive* statistical zero-knowledge proof systems for approximate SIVP and other lattice problems. Zero-knowledge proof systems are potentially useful building blocks both in the context of key registration in a public-key infrastructure (PKI), and in the construction of identification (ID) protocols. Finally, more efficient identification protocols (than those obtainable from zero-knowledge) were recently discovered by Lyubashevsky [42]. Remarkably, the proof systems of [42] are not zero-knowledge, and still they achieve secure identification under active attacks using an interesting aborting technique.

8 Open Questions

- **Cryptanalysis:** The experiments of [16] are very useful to gain some insight into the concrete hardness of lattice problems for specific values of the lattice dimension, as needed by lattice-based cryptography. But more work is still needed to increase our confidence and understanding, and in order to support widespread use of lattice-based cryptography. An interesting recent effort in this direction is the "Lattice Challenge" web page created by Lindner and Rückert [40, 10], containing a collection of randomly chosen lattices in increasing dimension for which finding short vectors is apparently hard.
- **Improved cryptosystems:** The LWE-based cryptosystem described in Section 5.4 is reasonably efficient and has a security proof based on a worst-case connection. Still, one might hope to considerably improve the efficiency, and in particular the public key size, by using structured lattices such as cyclic lattices. Another desirable improvement is to obtain a classical (i.e., non-quantum) worst-case connection. Finally, obtaining practical CCA-secure cryptosystems in the standard model is another important open question.
- **Comparison with number theoretic cryptography:** Can one factor integers or compute discrete logarithms using an oracle that solves, say, $\sqrt{n}$-approximate SVP? Such a result would prove that the security of lattice-based cryptosystems is superior to that of traditional number-theoretic-based cryptosystems (see [75, 1] for related work).

Acknowledgements

We thank Phong Ngyuen and Markus Rückert for helpful discussions on the practical security of lattice-based cryptography. We also thank Richard Lindner, Vadim Lyubashevsky, and Chris Peikert for comments on an earlier version.

References

[1] L. M. Adleman. Factoring and lattice reduction, 1995. Unpublished manuscript.

[2] D. Aharonov and O. Regev. Lattice problems in NP intersect coNP. *Journal of the ACM*, 52(5):749–765, 2005. Preliminary version in FOCS 2004.

[3] M. Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract) 10-19. In *Proc. 30th ACM Symp. on Theory of Computing (STOC)*, pages 10–19. ACM, 1998.

[4] M. Ajtai. Generating hard instances of lattice problems. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 1–32. Dept. Math., Seconda Univ. Napoli, Caserta, 2004. Preliminary version in STOC 1996.

[5] M. Ajtai. Representing hard lattices with $O(n \log n)$ bits. In *Proc. 37th Annual ACM Symp. on Theory of Computing (STOC)*, 2005.

[6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th Annual ACM Symp. on Theory of Computing (STOC)*, pages 284–293, 1997.

[7] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 601–610, 2001.

[8] L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[9] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003. Preliminary version in STOC'00.

[10] J. Buchmann, R. Lindner, and M. Rückert. Creating a lattice challenge, 2008. Manuscript.

[11] J.-Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proc. 38th IEEE Symp. on Found. of Comp. Science*, pages 468–477, 1997.

[12] J.-Y. Cai and A. Nerurkar. Approximating the SVP to within a factor $(1 + 1/\dim^{\varepsilon})$ is NP-hard under randomized reductions. *J. Comput. System Sci.*, 59(2):221–239, 1999.

[13] D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proc. of Eurocrypt '97*, volume 1233 of *LNCS*. IACR, Springer, 1997.

[14] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

[15] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 207–216, 2008.

[16] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *Advances in Cryptology – Proc. Eurocrypt '08*, Lecture Notes in Computer Science. Springer, 2008.

[17] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 197–206, 2008.

[18] C. Gentry and M. Szydlo. Cryptanalysis of the revised NTRU signature scheme. In *Proc. of Eurocrypt '02*, volume 2332 of *LNCS*. Springer-Verlag, 2002.

[19] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000. Preliminary version in STOC 1998.

[20] O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Technical Report TR96-056, Electronic Colloquium on Computational Complexity (ECCC), 1996.

[21] O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In *Advances in cryptology*, volume 1294 of *Lecture Notes in Comput. Sci.*, pages 105–111. Springer, 1997.

[22] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in cryptology*, volume 1294 of *Lecture Notes in Comput. Sci.*, pages 112–131. Springer, 1997.

[23] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sience*, 28(2):270–299, 1984. Preliminary version in Proc. of STOC 1982.

[24] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Computing*, 17(2):281–308, 1987.

[25] I. Haviv and O. Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, pages 469–477, 2007.

[26] J. Hoffstein, N. A. H. Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *Proc. of CT-RSA*, volume 2612 of *Lecture Notes in Comput. Sci.*, pages 122–140. Springer-Verlag, 2003.

[27] J. Hoffstein, N. A. H. Graham, J. Pipher, J. H. Silverman, and W. Whyte. Performances improvements and a baseline parameter generation algorithm for NTRUsign. In *Proc. of Workshop on Mathematical Problems and Techniques in Cryptology*, pages 99–126. CRM, 2005.

[28] J. Hoffstein, N. Howgrave-Graham, J. Pipher, and J. H. Silverman. Hybrid lattice reduction and meet in the middle resistant parameter selection for NTRUEncrypt. Submission/contribution to ieee p1363.1, NTRU Cryptosystems, Inc., URL http://grouper.ieee.org/groups/1363/lattPK/submissions.html#2007-02, 2007.

[29] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In *Proceedings of ANTS-III*, volume 1423 of *LNCS*, pages 267–288. Springer, June 1998.

[30] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Advances in cryptology (CRYPTO)*, pages 150–169, 2007.

[31] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. 15th ACM Symp. on Theory of Computing (STOC)*, pages 193–206. ACM, 1983.

[32] A. Kawachi, K. Tanaka, and K. Xagawa. Multi-bit cryptosystems based on lattice problems. In *Public Key Cryptography – PKC 2007*, volume 4450 of *Lecture Notes in Comput. Sci.*, pages 315–329, Berlin, 2007. Springer.

[33] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 126–135, 2004.

[34] S. Khot. Inapproximability results for computational problems on lattices, 2007. Survey paper prepared for the LLL+25 conference. To appear.

[35] P. Klein. Finding the closest lattice vector when it's unusually close. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 937–941, 2000.

[36] R. Kumar and D. Sivakumar. Complexity of SVP – a reader's digest. *SIGACT News*, 32(3):40–52, 2001.

[37] J. C. Lagarias, H. W. Lenstra, Jr., and C.-P. Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[38] A. K. Lenstra and H. W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.

[39] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[40] R. Lindner and M. Rückert. The lattice challence, 2008. Available at `http://www.latticechallenge.org/`.

[41] C. Ludwig. A faster lattice reduction method using quantum search. In *ISAAC*, pages 199–208, 2003.

[42] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC'08*, number 4939 in LNCS, pages 162–179, 2008.

[43] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2006.

[44] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *Fifth Theory of Cryptography Conference (TCC)*, volume 4948 of *Lecture Notes in Computer Science*. Springer, 2008.

[45] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: a modest proposal for FFT hashing. In *FSE 2008*, 2008.

[46] R. McEliece. A public-key cryptosystem based on algebraic number theory. Technical report, Jet Propulsion Laboratory, 1978. DSN Progress Report 42-44.

[47] D. Micciancio. Improving lattice based cryptosystems using the hermite normal form. In J. Silverman, editor, *Cryptography and Lattices Conference — CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 126–145, Providence, Rhode Island, Mar. 2001. Springer-Verlag.

[48] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM J. on Computing*, 30(6):2008–2035, Mar. 2001. Preliminary version in FOCS 1998.

[49] D. Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *Proc. 34th ACM Symp. on Theory of Computing (STOC)*, pages 609–618, 2002.

[50] D. Micciancio. Lattices in cryptography and cryptanalysis, 2002. Lecture notes of a course given in UC San Diego.

[51] D. Micciancio. Cryptographic functions from worst-case complexity assumptions, 2007. Survey paper prepared for the LLL+25 conference. To appear.

[52] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary versions in FOCS 2002 and ECCC TR04-095.

[53] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[54] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 372–381, 2004.

[55] D. Micciancio and S. Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In *Advances in cryptology (CRYPTO)*, volume 2729 of *Lecture Notes in Computer Science*, pages 282–298. Springer-Verlag, 2003.

[56] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21st ACM Symp. on Theory of Computing (STOC)*, pages 33–43, 1989.

[57] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21st ACM Symp. on Theory of Computing (STOC)*, pages 33–43, 1989.

[58] P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *Advances in cryptology (CRYPTO)*, volume 1462 of *Lecture Notes in Comput. Sci.*, pages 223–242. Springer, 1998.

[59] P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In *The 25th International Cryptology Conference (Eurocrypt)*, pages 271–288, 2006.

[60] P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In J. H. Silverman, editor, *Cryptography and Lattices, International Conference (CaLC 2001)*, number 2146 in Lecture Notes in Computer Science, pages 146–180, 2001.

[61] P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2008. To appear.

[62] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *3rd Theory of Cryptography Conference (TCC)*, pages 145–166, 2006.

[63] C. Peikert and A. Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, pages 478–487, 2007.

[64] C. Peikert and V. Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *Advances in Cryptology (CRYPTO)*, LNCS. Springer, 2008.

[65] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology (CRYPTO)*, LNCS. Springer, 2008.

[66] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 187–196, 2008.

[67] C. J. Peikert. Limits on the hardness of lattice problems in ℓ_p norms. *Computational Complexity*, 2008. To appear. Preliminary version in Proc. of CCC 2007.

[68] O. Regev. Lattices in computer science, 2004. Lecture notes of a course given in Tel Aviv University.

[69] O. Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in STOC'03.

[70] O. Regev. Quantum computation and lattice problems. *SIAM J. on Computing*, 33(3):738–760, 2004. Preliminary version in FOCS'02.

[71] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. 37th ACM Symp. on Theory of Computing (STOC)*, pages 84–93, 2005.

[72] O. Regev. Lattice-based cryptography. In *Advances in cryptology (CRYPTO)*, pages 131–141, 2006.

[73] O. Regev. On the complexity of lattice problems with polynomial approximation factors, 2007. Survey paper prepared for the LLL+25 conference. To appear.

[74] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2-3):201–224, 1987.

[75] C.-P. Schnorr. Factoring integers and computing discrete logarithms via Diophantine approximation. In J.-Y. Cai, editor, *Advances in computational complexity*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 171–182. AMS, 1993. Preliminary version in Eurocrypt '91.

[76] V. Shoup. NTL: A library for doing number theory. Available at `http://www.shoup.net/ntl/`.

[77] M. Szydlo. Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In *Proc. of Eurocrypt '03*, volume 2656 of *LNCS*. Springer-Verlag, 2003.

[78] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, University of Amsterdam, Department of Mathematics, Netherlands, 1981. Technical Report 8104.

[79] D. Wagner. A generalized birthday problem. In *Advances in cryptology (CRYPTO)*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.

Index